On the Accuracy and Complexity of Rate-Distortion Models for Fine-Grained Scalable Video Sequences

ChengHsin Hsu and Mohamed Hefeeda
School of Computing Science
Simon Fraser University
Surrey, Canada
{cha16, mhefeeda}@cs.sfu.ca

Technical Report: TR 2006-12

Abstract

Rate-distortion (R-D) models are functions that describe the relationship between the bitrate and expected level of distortion in the reconstructed video stream. R-D models enable optimization of the received video quality in different network conditions. Several R-D models have been proposed for, the increasingly becoming popular, fine-grained scalable video sequences. However, the models' relative performance has not been thoroughly analyzed. Moreover, the time complexity of each model is not known, nor is the range of bitrates in which the model produces valid results. This lack of quantitative performance analysis makes it difficult to select the model that best-suits a target streaming system. In this paper, we classify, analyze, and rigorously evaluate all R-D models proposed for FGS coders in the literature. We classify R-D models into three categories: analytic, empirical, and semi-analytic. We describe the characteristics of each category. We analyze the R-D models by following their mathematical derivations, scrutinizing the assumptions made, and explaining when the assumptions fail and why. In addition, we implement all R-D models, a total of eight, and evaluate them using a diverse set of video sequences. In our evaluation, we consider various source characteristics, diverse channel conditions, different encoding/decoding parameters, different frame types, and several performance metrics including accuracy, range of applicability, and time complexity of each model. We also present clear systematic ways (pseudo codes) for constructing various R-D models from a given video sequence. Based on our experimental results, we present a justified list of recommendations on selecting the best R-D models for video-on-demand, video conferencing, real-time, and peer-to-peer streaming systems.

I. Introduction

Video streaming on the Internet is increasingly getting very popular. The best-effort service offered by the Internet, however, poses unique challenges for high-quality video streaming. These challenges include heterogeneity and bandwidth variability in network channels between streaming servers and clients. These challenges require streaming systems to support bitrate scalability and error resiliency. Traditional streaming systems partially cope with these challenges using either multi-layer or multi-description encoding of streams. These solutions, however, provide limited (coarse-grain) rate scalability: clients receiving incomplete layers or descriptions cannot use them to enhance display quality. These solutions also suffer from poor error resiliency, because the loss or corruption of a few bits render the entire layer useless.

In contrast to traditional multi-layer video coding, fine granularity scalability (FGS) coding has been proposed to provide finer bitrate scalability and better error resiliency [1], [2]. An FGS encoder compresses video data into two layers: a base layer which provides basic quality, and a single enhancement layer that adds incremental quality refinements proportional to the number of bits received. Arbitrary truncation (at the bit level) of the enhancement layer to achieve a target bitrate is possible, and more importantly, it does not require complex or resource-intensive operations from the streaming servers or their proxy caches. This in turn enables streaming servers to scale to larger and more heterogeneous sets of clients.

Given the flexibility of controlling the bitrate provided by FGS encoders and the constraints on and the variability of the channel bandwidth, researchers seek to optimize the quality of received video streams. A common method in the literature to achieve such quality-optimized systems is through the use of rate-distortion (R-D) models. R-D models are functions that describe the relationship between the bitrate and expected level of distortion in the reconstructed video stream. Knowing the R-D models enables us, for example, to determine the required bitrate to achieve a target quality, to optimally allocate a given bandwidth among frames, and to prioritize bits within the same frame. Clearly, the accuracy of the R-D models directly impacts the performance of streaming systems using them. In addition, the time cost of constructing various R-D models for a given sequence may prefer, or even dictate, a certain model over others. For example, in real-time streaming systems, building an R-D model should be fast enough to cope with the timing constraints of the video stream.

Due to the increasing importance and adoption of FGS coding systems, several studies have proposed R-D models for them [3]–[6]. Each of these studies conducted limited performance evaluation, just enough to show the merits of the proposed model. The relative performance of different R-D models has not yet been thoroughly analyzed. Moreover, the time complexity of each model is not known, nor is the range of bitrates in which the model produces valid results. In addition, previous studies do not provide enough specifications to enable implementing and using the proposed R-D models. Finally, the importance of R-D models stems from their usefulness for different streaming systems. Because of the lack of quantitative performance analysis of different R-D models, it is currently difficult to select the model that best-suits a target streaming system.

In this paper, we classify, analyze, and rigorously evaluate all R-D models proposed for FGS coders that we are aware of. We classify R-D models into three categories: analytic, empirical, and semi-analytic. Analytic models abstract the characteristics of an input video sequence by the probability distribution of its DCT coefficients. This distribution is then used in mathematical equations that describe the encoding/decoding processes. To derive the final R-D equation, typically several simplifying assumptions are made, which sometimes compromise the accuracy of the analytic models. Empirical R-D models directly measure the actual distortion by decoding the video sequence at many sampling bitrates. We call the third category semi-analytic models because they are *inspired* by analytic models, but they do not use mathematical derivations to develop R-D models. Rather, each semi-analytic model proposes a parametrized function that is thought to approximate the actual R-D function. The parametrized function takes the shape of an analytically-derived function, but in a much simpler form. The parameters of the function are estimated using curve-fitting from a few actual rate-distortion data points.

We analyze the R-D models by following their mathematical derivations, scrutinizing the assumptions made, and explaining when the assumptions fail and why. In addition, we implement all R-D models, a total of eight, and evaluate them using a large and diverse set of carefully-chosen video sequences. In our evaluation, we consider various source characteristics, diverse channel conditions, different encoding/decoding parameters, different frame types, and several performance metrics including accuracy, range of applicability, and time complexity of each model. We also present clear systematic ways (pseudo codes) for constructing various R-D models from a given video sequence. Finally, our experimental results enable us to provide guidelines on selecting the most suitable R-D model for a target streaming system. We present a justified list of recommendations on choosing the best R-D models for video-on-demand, video conferencing, real-time, and peer-to-peer streaming systems.

The rest of this paper is organized as follows. In Section II, we review previous works on R-D modeling.

In Section III, we provide a brief background on video coding systems, reviewing the main concepts of non-scalable, layered, and fine grained scalable coders. We also describe how distortion is measured, and provide an overview on how analytic rate-distortion models are derived. Section IV presents and analyzes various analytic R-D models, and Section V describes empirical and semi-analytic models. Our evaluation study is presented in Section VI. We conclude the paper in Section VII, where we also present a list of recommendations on choosing the best R-D model for a target streaming system.

II. Related Work

Many R-D models have been proposed for nonscalable and layered coders. The pioneering work by Hang and Chen [7] analyzes a simple coder that consists of a uniform quantizer followed by an ideal entropy coder. The authors propose a simple analytic R-D model, and a parameterized R-D model that considers some characteristics of real coders. Although these model provide useful theoretical insights, the experimental study in [8] indicates that they are not very accurate. More recent R-D models use refinements based on observations from actual coders. For instance, the ρ-domain model [9], [10] leverages the high correlation between bitrates and percentage of nonzero-quantized coefficients. The ρ-domain model calculates distortion directly from raw DCT coefficients without constructing any distortion-quantization relationship.

To achieve higher accuracy, empirical R-D models have been proposed, e.g., in [11], [12], and [8]. In [11], a rate-quantization model is proposed as $R(\Delta) = \alpha + (\beta/\Delta^{\gamma})$, where α, β, and γ are estimated using actual measurements. In [8], a piecewise cubic R-D model is proposed. The R-D model in [12] utilizes the high correlation of R-D curves among consecutive frames by using the quadratic relationship: $R(D) = aD^{-1} + bD^{-2}$, where a and b are estimated using empirical samples from previous frames.

All of the above models were proposed for nonscalable and layered coders that are quite different from the recent FGS coders. Various studies have shown that these models are not directly applicable to FGS coders. For example, [3] investigates the accuracy of the uniform quantization model of [7] and the quadratic model of [12] when applied to FGS-encoded sequences. The experiments indicate that these two models are not applicable to FGS coders. In addition, Zhang et al. [13] experimentally show that the piecewise cubic model is inaccurate for FGS coders. Furthermore, because the ρ-domain model has no distortion-quantization model, applying ρ-domain model to FGS coders requires bitplane truncations and direct distortion computations. This process is similar to pure empirical approaches, which has high computational complexity. Therefore, the ρ-domain model may not be applicable to FGS coders.

Recently several studies have proposed R-D models for FGS coders. Most of these models are inspired by the R-D models of nonscalable and layered coders described above. For example, the square root model [3] for FGS coders is a generalization of the R-D model in [7]. The linear rate-quantization function purposed in the ρ-domain model [10] is employed in the FGS logarithm model [4]. The piecewise cubic model [8] is transformed into the FGS piecewise linear model [13].

To the best of our knowledge, the performance and complexity of FGS R-D models have not been *rigorously* studied before. Only limited comparisons among some of the models were conducted. In [3], [4], the authors compared their new R-D models against nonscalable and layered R-D models for accuracy. They analyzed four sequences that are encoded with fixed coding parameters. More importantly, the comparison was conducted only at bitplane boundaries. Since FGS-encoded streams can be truncated at arbitrary bit positions, the accuracy of the R-D models across all bitrates should be considered. In [5], the presented model was not compared against other models. Instead, different curve fitting schemes of the proposed model were compared. Similarly, the study in [13] only compared two alternatives: piecewise linear and piecewise exponential models.

We believe that our comparative study is rigorous because: (i) We compare the FGS R-D models against each other; (ii) We use a rich set of performance metrics for accuracy, applicable bitrate range, and time complexity; (iii) We choose wide ranges of sampling bitrates that match common usage patterns of various streaming applications; and (iv) We use a diverse set of video sequences encoded with various parameters.

III. Background

In this section, we provide a brief background on video coding systems, reviewing the main concepts of non-scalable, layered, and fine grained scalable coders. We also describe how distortion is measured, and provide an overview on how analytic rate-distortion models are derived.

A. Non-scalable and Layered Coding Systems

In video coding, each frame is first divided into non-overlapping blocks. These blocks are then passed through a transformer for energy concentration and coefficient de-correlation. Energy concentration reduces the number of coefficients, which leads to a higher compression rate. The coefficient de-correlation saves resources by allowing processing of coefficients one-by-one, i.e., scalar quantization, without much coding efficiency penalty. The coefficients are then quantized, such that a continuous set of inputs can be represented by a finite set of discrete values. The main purpose of quantization is to reduce unnecessary details. A quantizer is defined with its quantization bins and corresponding reconstruction values. At the encoder side, inputs falling within the range of a quantization bin are mapped to the same quantization index of that bin. The quantization index is a small natural number that is sent through a communication channel to the decoder. At the decoder side, each quantization index is mapped back to the reconstruction value that corresponds to that index. A quantizer with fixed bin size is called a uniform quantizer, where the bin size is called the quantization step. In uniform quantizers, larger quantization steps mean higher compression rates and more distortion. Most video coding standards employ a quantization parameter to scale the bin size, which is typically proportional to the quantization step.

Non-scalable coding systems optimize coded streams at a single bitrate. This implies that clients have to meet a minimum bandwidth requirement to receive and decode the stream. Meanwhile, clients with higher bandwidths cannot get better quality from the coded bitstream. To address this issue, layered bitrate scalable coding systems have been proposed. In layered scalable coding systems, data is divided into a base layer to provide the basic decoded output, and one or more enhancement layers to provide quality refinements. Receivers have the flexibility to subscribe and process as many enhancement layers as they wish to make use of the additional bandwidth they might have.

Layered scalability can be realized by different means [14, Section 11.1]. For instance, we can code the base layer at a lower frame rate, and incrementally put more uncoded frames into successive enhancement layers to achieve temporal scalability. Signal-to-Noise (SNR) scalability is another example, where each enhancement layer uses a finer (smaller) quantization parameter to encode the quantization error of previous layers. These scalable coders can simultaneously deliver video streams at more than one bitrates.

Layered coding systems are usually called coarse-grained scalable coding systems, because an enhancement layer is not decodable unless all bits in it are received. In addition, number of layers is typically very small because of coding complexities and layering overhead. Therefore, layered scalable coding systems provide limited flexibility for streaming applications.

B. Fine Granularity Coding Systems

Video streaming over a dynamic and diverse environment like the Internet requires greater flexibility than that is provided by layered coding systems. Therefore, fine granularity scalability (FGS) has been designed to cover a wide range of bitrates at fine (bit level) steps. An FGS coding system encodes the video into two layers: a base layer and an enhancement layer. The base layer supports a single bitrate that delivers the basic video quality. The enhancement layer improves upon the quality of the base layer with a gain that is proportional to the number of *bits* received.

An FGS coding system may, in general, adopt any transform coders for its base layer and any FGS technique for its enhancement layer. Because of its simplicity and good coding efficiency [1], [2], the bitplane coding defined in the MPEG-4 standard is the most recognized FGS technique nowadays. The basic idea of bitplane coding is to represent DCT coefficients as binary digits. Binary digits at the same significant position form a bitplane. Bitplanes are ordered by their significance, where a more significant bitplane contains more information per bit. Losing a bit in a higher significant bitplane results in a larger distortion than losing a bit in a lower significant bitplane. Each bitplane is separately run-length coded using the symbol format (RUN, LAST). These symbols are then coded by a variable length coder (VLC) into a bitstream [15].

To illustrate how bitplane coding works, we present a simple example. This example also draws analogies between bitplane coding and uniform quantization. Both techniques are used to meet resource constraints by dropping less important details.

In bitplane coding, each coefficient is coded individually, therefore, we concentrate on a single coefficient for simplicity. Table I shows that we need $Z = 4$ bitplanes to code an original coefficient of value 15 (0x0F) in its binary format. Suppose the FGS decoder decodes $z \leq Z$ bitplanes. For example, if $z = 2$, the decoder decodes the first two significant bitplanes, and the reconstructed level is 12 (0x0C) instead of 15. Clearly, the reconstruction level is a function of z: We get no information if $z = 0$; and we get a perfect reconstruction if $z = Z$.

Notice that bitplane coding is essentially a quantizer with a uniform quantization step: $\Delta(z) = 2^{(Z-z)}$ (see the last two rows of Table I). This is because the information in the last $(Z - z)$ bitplanes is not seen by the decoder, as if this information were lost during quantization with a step of $2^{(Z-z)}$. We also observe that the reconstruction levels are defined by masking out the last $(Z - z)$ bitplanes. Hence, we can write the reconstruction level of an input coefficient x as:

$$L(x) = \begin{cases} \lfloor x/\Delta \rfloor \times \Delta, & x \geq 0; \\ \lceil x/\Delta \rceil \times \Delta, & x < 0, \end{cases} \tag{1}$$

where Δ is the quantization step.

FGS coding provides three major advantages to streaming applications: (1) better support for heterogeneous environments through bit-level scalability; (2) better error resiliency because partially corrupted bitstreams can be decoded; and (3) better streaming server scalability due to the separation of encoding and streaming processes.

To avoid propagating corrupted information to other frames, an FGS coder does not use enhancement layer refinements for motion compensations. While this approach provides extra error resiliency, motion

TABLE I

BITPLANE CODING EXAMPLE.

	Original	**Reconstructed**				
DCT Coefficient	0x0F	0x00	0x08	0x0C	0x0E	0x0F
Bp #1	1	0	1	1	1	1
Bp #2	1	0	0	1	1	1
Bp #3	1	0	0	0	1	1
Bp #4	1	0	0	0	0	1
Quantization step, Δ		2^4	2^3	2^2	2^1	2^0
#decoded bitplanes, z		0	1	2	3	4

estimating using the base layer reconstruction produces higher errors thus negatively impacts the coding efficiency. Progressive fine granularity scalability (PFGS) has been proposed for better coding efficiency by utilizing part of the enhancement layer for motion compensation [16].

Researchers have noticed the low coding efficiency in the less significant bitplanes. This can be explained by the uniformly distributed run-length symbols found in them. Several solutions have been proposed, for example, [17] proposes to divide bits into two groups: the significant bits and refinement bits. The former group is coded with run-length and entropy coders, in contrast, the latter group is coded with only entropy coder. A 0.25 dB Peak Signal-to-Noise ration improvement is reported by skipping the run-length coding in the group of refinement bits. Based on this idea, Chao et al. proposed a three group approach [18].

C. Measuring Distortion and Rate-Distortion (R-D) Models

In the literature, distortion is commonly measured in terms of the mean square error (MSE) between the luminance values of pixels in the original and reconstructed frames. Chrominance components are usually ignored, because the human visual system is less sensitive to them compared to the luminance components, and they only occupy about 10% of the bitrate [19]. A related quality measure is the Peak Signal-to-Noise Ratio (PSNR), which is given by: $\text{PSNR} = 10 \log_{10} \frac{255^2}{\text{MSE}}$ dB. PSNR is used in video quality comparison.

Distortion is clearly related to the quantization step Δ, because reconstruction levels depend on Δ according to (1). In addition, in the previous section we established the similarity between uniform quantization in traditional coding and bitplane coding. Hence, following [7], we can write the distortion (in MSE) as:

$$D(\Delta) = 2\sum_{i=0}^{N} \int_{i\Delta}^{(i+1)\Delta} f(x)(x - i\Delta)^2 dx, \tag{2}$$

where $f(x)$ is a symmetric probability density function describing the distribution of the DCT coefficients. Intuitively, the above equation can be understood as follows. The integral computes the distortion in a given quantization bin. Since $i\Delta$ is the reconstruction level for any point x within the range of the $[i\Delta, (i+1)\Delta]$ bin, $(x - i\Delta)^2$ represents the square of the error that occurs if a DCT coefficient takes on the value x. The probability that a coefficient takes on the value x is described by $f(x)$. Hence, multiplying $(x - i\Delta)^2$ by $f(x)$ and integrating over the range of a bin yield the distortion (in MSE) in that bin. The summation aggregates the distortion from all bins. Notice that the summation iterates only over one-half of the total quantization bins, i.e., over N out of the $2N$ bins. This is because of the symmetry of $f(x)$.

We call (2) the basic distortion-quantization (D-Q) function, because: (i) it relates the quantization step with the expected distortion, and (ii) it is typically the start point for developing analytic rate-distortion models.

Rate-distortion (R-D) models are functions that predict the expected distortion at a given bitrate. This is important for streaming applications that strive to optimize rendered quality in environments where channel conditions vary dynamically. R-D models are derived in three steps. First, a probability density function $f(x)$ is assumed for the distribution of DCT coefficients. This density function is called the source model. The assumed density function is then substituted in (2), which yields—after approximation and manipulation—the D-Q function of the model. The second step in deriving R-D models is to derive a relationship between the bitrate and the quantization step. This is based on insights from the encoding/decoding processes. This relationship is called the R-Q function. Finally, the D-Q and R-Q functions are solved together to obtain the R-D model. We present and evaluate several R-D models in the next three Sections.

IV. Analytic Rate-Distortion Models

Rate-distortion (R-D) models are functions that describe the relationship between bitrate and expected level of distortion in a reconstructed video stream. In this section, we present three analytic R-D models that are explicitly designed for the FGS enhancement layer. For each analytic model, we present the adopted source model. Then, we derive the R-D model in three steps: (i) we derive a distortion-quantization (D-Q) function, which relates the quantization step with the expected distortion; (ii) we derive a rate-quantization (R-Q) function, which relates the quantization step with the bitrate of the resulting bitstream; and (iii) we solve the D-Q and R-Q functions together either analytically or numerically to obtain the R-D function. Finally, we present our own implementation of each model. In Section VI, we experimentally evaluate and compare these R-D models.

A. *Square Root Model*

The square root model assumes that DCT coefficients follow a two-component Laplacian mixture (LM) distribution [3]. A Laplacian mixture function is a linear combination of several Laplacian density functions. A two-component Laplacian mixture function is given as: $f_{LM}(x) = p_1 \frac{\lambda_1}{2} e^{-\lambda_1 |x|} + p_2 \frac{\lambda_2}{2} e^{-\lambda_2 |x|}$, where p_1, p_2 and λ_1, λ_2 are parameters that need to be estimated. Like other finite mixture distributions, the parameters of Laplacian mixture are not easily estimated using log-likelihood functions. Instead, researchers adopt numerical methods such as the expectation-maximization (EM) method [20, Section 8.4]. EM methods are widely used for: (1) incomplete observations, and (2) finite mixture models. We have designed and implemented an EM estimator based on the multi-dimensional estimator for Laplacian mixture distributions proposed in [21]. The details of this estimator are given in [22].

To derive a D-Q function, the square root model simplifies the basic D-Q function in (2) using the *high-resolution hypothesis*. The high-resolution hypothesis states that if bin sizes are adequately small relative to the variation rate of the density function, the density function—in each bin—can be approximated by a uniform density [23]. This hypothesis leads to a new D-Q function:

$$D(\Delta) = 2 \sum_{m=0}^{N} \sum_{n=m\Delta}^{(m+1)\Delta-1} (n - m\Delta)^2 f_{LM}(n). \tag{3}$$

Unlike (2), the D-Q function in (3) uses a series of sub-bins to approximate the integral, and it assumes a constant density function in each sub-bin. After substituting the LM source model, we get:

$$\begin{aligned} D(\Delta) = &\frac{p_1}{(e^{-\lambda_1 \Delta} - 1)} \left(e^{-\lambda_1 (\Delta-1)} [(\Delta - 1 + \frac{1}{\lambda_1})^2 + \frac{1}{{\lambda_1}^2}] - \frac{2}{{\lambda_1}^2} \right) + \\ &\frac{p_2}{(e^{-\lambda_2 \Delta} - 1)} \left(e^{-\lambda_2 (\Delta-1)} [(\Delta - 1 + \frac{1}{\lambda_2})^2 + \frac{1}{{\lambda_2}^2}] - \frac{2}{{\lambda_2}^2} \right), \end{aligned} \tag{4}$$

where p_1, p_2 and λ_1, λ_2 are Laplacian mixture parameters.

We make two observations on this D-Q function. First, the high-resolution hypothesis introduces higher approximation errors when the variation rate of the density function is large. During our experiments (Section VI), we observed that the DCT density function has high variability in two cases: (i) when the base layer is encoded at high bitrate, and (ii) when the video sequence has low temporal and spatial complexity. In both cases, the accuracy of the square root model was worse than other models. The second observation is that the square root model greatly under-estimates the distortion for small quantization steps.

For example, setting $\Delta = 1$ in (3), yields $D(1) = 2\sum_{m=0}^{N}\sum_{n=m}^{m}(n-m)^2 f_{LM}(n) = 0$. Consequently, the produced PSNR approaches infinity as Δ goes to 1. This results in exaggerated (infinity) estimation of the stream quality, which may mislead streaming applications using this model.

To derive the R-Q function for this model, the bitrate is first expressed as a function of number of decoded bitplanes z. Then z is replaced by the quantization step Δ according to the relationship derived in the previous section: $\Delta = 2^{Z-z}$, where Z is the total number of bitplanes. The square root model proposes the following quadratic $R(z)$ function:

$$R(z) = c_1 z^2 + c_2 z + c_3, \tag{5}$$

where c_1, c_2, c_3 are parameters that need to be estimated. To justify this function, Dai et al. empirically show that second order polynomials are sufficient for typical $R(z)$ functions [3]. Furthermore, they prove that an actual $R(z)$ function changes convexity up to once, thus a second order polynomial is enough [6]. The proof, however, assumes that DCT coefficients follow a Laplacian, not Laplacian mixture, distribution. To estimate c_1, c_2, c_3, a set of (R, z) values is needed. This set is easily computed at bitplane boundaries: At each bitplane boundary $z = 1, 2, \ldots, Z$, the bitrate is computed as $\sum_{i=1}^{z} l_i/T$, where l_i is the size of bitplane i and T is the frame period.

Given that $\Delta = 2^{Z-z}$, we get the R-Q function:

$$R(\Delta) = c_1(Z - \log_2 \Delta)^2 + c_2(Z - \log_2 \Delta) + c_3. \tag{6}$$

Computing Δ from (6) and substituting (6) in (4), we obtain the $D(R)$ function of the square root model. The resulting equation is fairly complicated, therefore, not shown here.

We have implemented the square root model as follows. First, we estimate the $p_1, p_2, \lambda_1, \lambda_2$ parameters of the Laplacian mixture density function from the DCT coefficients of the considered frame. We use an expectation-maximization estimator. Then, we extract the size of each bitplane in the enhancement layer of the frame. Using these sizes, we construct a set $\mathbb{R}_z$ of (R, z) elements at bitplane boundaries as described above. Then, we curve-fit the elements of $\mathbb{R}_z$ to (5) and obtain c_1, c_2, c_3. The square root R-D model is completely specified once we determine the source model parameters $p_1, p_2, \lambda_1, \lambda_2$ and the constants c_1, c_2, c_3. The pseudocode in Figure IV-A summarizes our implementation of the square root model.

B. Logarithm Model

The logarithm model employs a Laplacian mixture (LM) source model [4], same as the square root model. We use the same parameter estimator used for the square root model.

```
SqrtModel
/* Inputs:
*    FGS-encoded frame $f$
*    Frame period $T$ (in sec)
*/
/* Output:
*    Square root R-D model of the frame, specified by $p_1, p_2, \lambda_1, \lambda_2$ and $c_1, c_2, c_3$.
*/
1.  $< Z; l_1, l_2, \ldots, l_Z >=$ extractBitplaneInfo($f$);
2.  $\mathbb{C} =$ getDCTcoefficients($f$);
3.  Use expectation-maximization method to estimate $p_1, p_2, \lambda_1, \lambda_2$
    of the Laplacian mixture distribution using $\mathbb{C}$;
4.  $R = 0$; $\mathbb{R}_z = \{(0,0)\}$;
5.  for $i = 1$ to $Z$ do
6.    $R = R + l_i/T$;
7.    $\mathbb{R}_z = \mathbb{R}_z \cup \{(R, i)\}$;
8.  endfor
9.  Curve-fit elements of $\mathbb{R}_z$ to $R(z) = c_1 z^2 + c_2 z + c_3$ to determine $c_1, c_2, c_3$;
```

Fig. 1. Pseudo code for the square root R-D model.

To derive a D-Q function, the set of enhancement layer coefficients $\mathbb{C}$ is partitioned into two subsets: $\mathbb{C}_0$ and $\mathbb{C}_{\bar{0}}$. $\mathbb{C}_0$ contains coefficients falling in the interval $(-\Delta, \Delta)$, while $\mathbb{C}_{\bar{0}}$ contains all other coefficients. Note that the bitplane coding quantizes all coefficients in $\mathbb{C}_0$ to the quantization index with zero reconstruction level. Hence, $\mathbb{C}_0$ coefficients are called zero-quantized. We define $N = |\mathbb{C}|$ and $M = |\mathbb{C}_{\bar{0}}|$ to be the number of total and zero-quantized coefficients, respectively.

The study in [24] reveals that $\mathbb{C}_0$ coefficients are responsible for the majority of distortion at low and medium bitrates. Therefore, the logarithm model strives to accurately compute the distortion caused by $\mathbb{C}_0$, and to roughly approximate the distortion related to $\mathbb{C}_{\bar{0}}$. We define D_0 and $D_{\bar{0}}$ as the distortion contributed by $\mathbb{C}_0$ and $\mathbb{C}_{\bar{0}}$, respectively. We separately analyze each of them as follows.

Since all coefficients in $\mathbb{C}_0$ have zero reconstruction level, D_0 is given by: $D_0(\Delta) = \sum\limits_{c_i \in \mathbb{C}_0} |c_i|^2$. On the other hand, $D_{\bar{0}}$ is approximated as: $D_{\bar{0}}(\Delta) = M\Delta^2/12$, where $\Delta^2/12$ is the average distortion of a single coefficient if a uniform quantizer is employed under the high resolution hypothesis. The total distortion D is computed as the normalized sum of D_0 and $D_{\bar{0}}$:

$$D(\Delta) = \frac{D_0 + D_{\bar{0}}}{N} = \frac{1}{N} \sum_{c_i \in \mathbb{C}_0} |c_i|^2 + \frac{M}{N} \frac{\Delta^2}{12}. \tag{7}$$

Although the total number of coefficients N is constant, the number of coefficients in $\mathbb{C}_{\bar{0}}$ is a function of Δ. We define ζ as the percentage of nonzero-quantized coefficients. That is, $\zeta = M/N$, and it is a function of Δ. Observe that ζ is the probability that DCT coefficients do not fall in the range $(-\Delta, \Delta)$. Thus, a Laplacian mixture density function, ζ is given by:

$$\begin{aligned} \zeta = M/N =& 1 - \int_{-\Delta}^{\Delta} (p_1 \frac{\lambda_1}{2} e^{-\lambda_1 |x|} + p_2 \frac{\lambda_2}{2} e^{-\lambda_2 |x|}) dx \\ =& 1 - (p_1 e^{-\lambda_1 \Delta} + p_2 e^{-\lambda_2 \Delta}), \end{aligned} \tag{8}$$

where p_1, p_2, λ_1, λ_2 are parameters of the density function. Substituting (8) in (7), the D-Q function is obtained.

The logarithm model utilizes a linear R-Q function that was proposed for nonscalable coders in [25]. The applicability of this linear R-Q function to fine-grained scalable coders was validated through simulations in [4]. The linear R-Q model states that the bitrate is a linear function of the percentage of nonzero-quantized coefficient, that is:

$$R = \gamma \zeta = \gamma - \gamma (p_1 e^{-\lambda_1 \Delta} + p_2 e^{-\lambda_2 \Delta}), \tag{9}$$

where γ is the slope parameter that needs to be estimated. To estimate γ, a set of (R, ζ) values is needed. This set is constructed at bitplane boundaries: At each bitplane boundary $z = 1, 2, \ldots Z$, the bitrate is computed as $\sum_{i=1}^{z} l_i / T$, and the percentage of nonzero-quantized coefficients is computed from the DCT coefficients using a quantization step of 2^{Z-z}.

Analytically solving the D-Q function in (7) and the R-Q function in (9) to obtain the R-D model is very complex, if at all possible. To overcome this complexity, the authors of the logarithm model proposed the following approximation [4]. They first solve (7) and (9) by assuming a single Laplacian distribution. Then a linear combination of the solution is used as an approximation for the Laplacian mixture case. Specifically, solving (7) and (9) using a Laplacian distribution yields:

$$D(R) = \frac{2}{\lambda^2} - \frac{11 \log (\gamma / R) + 24 \log (\gamma / R) + 24}{12 \lambda^2} \frac{R}{\gamma}. \tag{10}$$

And a linear combination of (10) results in the approximated $D(R)$ function for the logarithm model as:

$$\begin{aligned} D(R) &= a_1 - (a_2 \log^2 R + a_3 \log R + a_4)R, \text{ where} \\ a_1 &= \frac{2p_1}{{\lambda_1}^2} + \frac{2p_2}{{\lambda_2}^2}, \\ a_2 &= \frac{11p_1}{12{\lambda_1}^2\gamma} + \frac{11p_2}{12{\lambda_2}^2\gamma}, \\ a_3 &= \frac{(-24 - 22\log\gamma)p_1}{12{\lambda_1}^2\gamma} + \frac{(-24 - 22\log\gamma)p_2}{12{\lambda_2}^2\gamma}, \\ a_4 &= \frac{p_1(24 + 24\log\gamma + 11\log^2\gamma)}{12{\lambda_1}^2\gamma} + \frac{p_2(24 + 24\log\gamma + 11\log^2\gamma)}{12{\lambda_2}^2\gamma}. \end{aligned} \tag{11}$$

We have implemented the logarithm model as follows. Similar to the case of the square root model, we first estimate the $p_1, p_2, \lambda_1, \lambda_2$ parameters of the Laplacian mixture density using an expectation-maximization method. Then, we extract the size of each bitplane in the enhancement layer of the frame. Using these sizes and the set of coefficients $\mathbb{C}$, we construct a set $\mathbb{R}_\zeta$ of (R, ζ) elements at bitplane boundaries as described above. Then, we curve-fit the elements of $\mathbb{R}_\zeta$ to (9) and obtain γ. Last, we compute a_1, a_2, a_3, a_4 using (11). The logarithm root R-D model is completely specified once we determine the constants a_1, a_2, a_3, a_4. The pseudocode in Figure IV-B summarizes our implementation of the logarithm model.

C. Generalized Gaussian Function Model

The generalized Gaussian function (GGF) model uses 64 distributions—of the same family but with different parameters—as the source model [5]. Each distribution models the set of coefficients of the same frequency component, and leads to an R-Q and a D-Q function for that frequency component. Recall that a frame is divided into 8x8 pixel blocks, and the DCT is applied on each block. This results in a set of 64 DCT coefficients per block, each belonging to a different frequency. Combining 64 R-Q functions gives a frame-level R-Q function, and aggregating 64 D-Q functions results in a frame-level D-Q function.

Zero-mean generalized Gaussian functions have been used to model DCT coefficients in the literature [26]. Its density function is defined as:

$$f_{GGF}(x) = \frac{\nu\alpha(\nu)}{2\sigma\Gamma(1/\nu)} e^{-(\frac{\alpha(\nu)}{\sigma}|x|)^\nu}, \tag{12}$$

where $\alpha(\nu) = \sqrt{\frac{\Gamma(3/\nu)}{\Gamma(1/\nu)}}$, and $\Gamma(\cdot)$ denotes the Gamma function, which is defined as $\Gamma(z) = \int_0^\infty e^{-t}t^{z-1}dt$. $\nu, \sigma > 0$ are the shape and standard deviation parameters of the distribution. This distribution covers a

```
LogModel
/* Inputs:
*    FGS-encoded frame $f$
*    Frame period $T$ (in sec)
*/
/* Output:
*    Logarithm R-D model of the frame, specified by $a_1, a_2, a_3, a_4$.
*/
1.  $< Z; l_1, l_2, \ldots, l_Z >=$ extractBitplaneInfo($f$);
2.  $\mathbb{C} =$ getDCTcoefficients($f$);
3.  Use expectation-maximization method to estimate $p_1, p_2, \lambda_1, \lambda_2$
    of the Laplacian mixture distribution using $\mathbb{C}$;
4.  $R = 0$; $\mathbb{R}_\zeta = \{(0, 0)\}$;
5.  for $i = 1$ to $Z$ do
6.    $R = R + l_i/T$;
7.    $\mathbb{C}_\zeta =$ getQuantizedCoeff($\mathbb{C}, 2^{Z-i}$);
8.    $\zeta = |\mathbb{C}_\zeta|/|\mathbb{C}|$;
9.    $\mathbb{R}_\zeta = \mathbb{R}_\zeta \cup \{(R, \zeta)\}$;
10. endfor
11. Curve-fit elements of $\mathbb{R}_\zeta$ to $R(\zeta) = \gamma\zeta$ to determine $\gamma$;
12. Use Eq. (11) to compute $a_1, a_2, a_3, a_4$;
```

Fig. 2. Pseudo code for the logarithm R-D model.

wide range of probability distributions, as it degenerates to a Laplacian distribution when $\nu = 1$, and to a Gaussian distribution when $\nu = 2$.

The parameters of generalized Gaussian functions are often estimated using one of the two methods [27]: (i) a maximum likelihood method, and (ii) a moment method[1]. In the first method, a nonlinear likelihood equation for estimating ν, which was shown to be a maximum likelihood estimator if a unique root exists [29]. Furthermore, this likelihood equation has a unique root when the sample size approaches infinity. When we implemented this method, though, the equation did not have a unique root in some

[1]The same estimator is independently proposed in another work [28].

cases. This is because of the relatively small sample size of DCT coefficients. In contrary, the moment method constantly results in a single estimation. In addition, experimental results show that it only suffers marginal loss of accuracy [27]. Hence, we adopt the moment estimator for its simplicity and robustness. The details of this estimator are given in [22].

To derive a D-Q model, the GGF model assumes a zero-mean source distribution (denoted by f_{ZM}) and the high-resolution hypothesis. Sun et al. substitute $f_{ZM}(x)$ for $f(x)$ in Eq. (2); after manipulation and simplification, the D-Q model is given by [5]:

$$D(\Delta) = \Delta^2/3. \tag{13}$$

This $D(\Delta)$ is different from the classic uniform quantizer's $D(\Delta) = \Delta^2/12$ approximation, because of the different reconstruction levels: A bitplane coder uses the floor function rather than the bin midpoints for reconstructions (as discussed in Section III-B).

To derive its R-Q function, the GGF model assumes an ideal variable-length coder (VLC) whose coding efficiency can arbitrarily approach the source entropy. Therefore, the GGF model employs the source entropy to estimate the bitrate as:

$$R(\Delta) = -\int_{-\infty}^{\infty} f_{ZM}(x)\log_2 f_{ZM}(x)dx - \log_2 \Delta, \tag{14}$$

where the R is in bits per pixel. Notice that, this R-Q model imposes high computational complexity if the integral $\int_{-\infty}^{\infty} f_{ZM}(x)\log_2 f_{ZM}(x)dx$ is not analytically solvable. Unfortunately, replacing the generalized Gaussian function source model for f_{ZM} produces a function that requires a numerical integration.

The R-D function is derived by combining the D-Q model in (13) and the R-Q model in (14). We get $D_i(R_i)$ for component i as:

$$D_i(R_i) = 2^{-2R_i - 2\int_{-\infty}^{\infty} f_{GGF}(x)\log_2 f_{GGF}(x)dx}/3. \tag{15}$$

Next, we consider the aggregation of these 64 $D_i(R_i)$ functions for a frame-level $D(R)$ model. Note that a given bitrate R is achieved by truncating a coded stream. This truncation decides an R_i for each component i ($i = 1, 2, \ldots, 64$). These 64 components, however, are not independently coded. Therefore, decomposing R into R_i is non-trivial. To address this difficulty, authors of [5] propose an exhaustive search on the quantization step Δ from 1 to 64. For each Δ, a frame-level R-D mapping is computed by composing 64 D_i's (given by (13)) and 64 R_i's (given by (14)). We follow this approach to compute R-D estimates for Δ's. We then construct a piecewise linear R-D model using these estimations to cover intermediate Δ for completeness.

```
GGFModel
/* Inputs:
*    FGS-encoded frame $f$
*/
/* Output:
*    GGF R-D model of the frame, specified by a piecewise linear function that
*    connects elements of $\mathbb{D}_R$.
*/
1.  $< \mathbb{C}_1, \mathbb{C}_2, \ldots, \mathbb{C}_{64} >=$ getDCTcoefficients($f$);
2.  for $i = 1$ to $64$ do
3.    Use moment estimator to estimate $\nu_i, \sigma_i$ of the generalized Gaussian function
      using $\mathbb{C}_i$;
4.  endfor
5.  $\mathbb{D}_R = \{\}$;
6.  for $\Delta = 1$ to $64$ do
7.    $R = 0$;
8.    for $i = 1$ to $64$ do // frequencies
9.      $R = R +$ getComponentBitrate($\nu_i$, $\sigma_i$, $\Delta$); // Eq. (14)
10.   endfor
11.   $\mathbb{D}_R = \mathbb{D}_R \cup \{(\Delta^2/3, R)\}$;
12. endfor
```

Fig. 3. Pseudo code for the GGF R-D model.

We have implemented the GGF model as follows. We first estimate the ν_i, σ_i parameters of the generalized Gaussian function for frequency component i ($i = 1, 2, \ldots, 64$) using a moment estimator. Then, we compute a set $\mathbb{D}_R$ of (D, R) elements at integer quantization steps Δ from 1 to 64 as described above. The GGF R-D model is completely specified once we construct a piecewise linear function using the elements of $\mathbb{D}_R$. Because of the space limitations, we give the pseudocode of the GGF model in [30]. The pseudocode in Figure IV-C summarizes our implementation of the GGF model.

V. Empirical and Semi-analytic Models

In this section, we present two empirical methods: (i) a pure empirical model for determining accurate R-D functions which we use for comparisons, and (ii) an efficient piecewise linear model. Then, we present three semi-analytic models that are inspired by the three analytic models described in the previous section. Semi-analytic models do not use mathematical derivations to develop R-D models. Rather, each semi-analytic model proposes a parametrized function that is thought to approximate the actual R-D function. The parametrized function takes the shape of an analytically-derived function, but in a much simpler form. The parameters of the function are estimated using curve-fitting from a few actual rate-distortion data points.

A. Empirical Model

The pure empirical approach is a brute force method to derive R-D functions. It chooses many sampling bitrates and decodes the video sequence at each of them. Then, for each sample bitrate, it computes the distortion as the difference between the original and reconstructed video sequences. Interpolation is used to extend the discrete R-D mapping into a continuous function. The empirical approach requires tremendous amount of computational power and storage space. Nevertheless, it produces accurate R-D curves. We use results produced by the empirical approach as the baseline for comparing various R-D models. In our implementation of this empirical model, we choose six points in each bitplane, including the bitplane boundaries. And we compute the distortion at these points.

We note that constructing empirical R-D models for FGS-encoded sequences does *not* requires re-encoding of the sequence. We only need to truncate the bitstream at appropriate bit positions. This is in contrast to constructing empirical R-D models for non-scalable or layered-scalable sequences, which requires re-encoding a video sequence at each sampling bitrate. Re-encoding is time consuming because coding systems employ many optimization procedures at the encoding time. Therefore, while empirical R-D models for FGS-encoded sequences are expensive to construct, they are much less expensive, and more feasible than empirical R-D models for nonscalable or layered-scalable sequences.

B. Piecewise Linear Model

Traditional empirical R-D models often employ exponential interpolation between sampling points. Zhang et al. [13] found that exponential interpolations do not accurately track the actual R-D curves of FGS-encoded sequences. The experiments in [13] reveal that when sampling bitrates are located in the same bitplane, using linear interpolation produces smaller deviation than using exponential interpolation.

However, if the sampling bitrates are from different bitplanes, applying exponential interpolation is more accurate. To implement the piecewise linear model, we choose sampling points at all bitplane boundaries and compute the distortion at these points. Then we connect these points with line segments.

C. Semi-analytic Square Root Model

This model is based on the analytic square root model in Section IV-A. However, instead of using the complicated D-Q function in (4), it uses the following *heuristic* function:

$$PSNR(z) = d_1 z^2 + d_2 z + d_3, \tag{16}$$

where PSNR is the quality, not the distortion, and d_1, d_2, d_3 are parameters that need to be estimated. This function is justified by the fact that the quality (in terms of PSNR) is a monotonically increasing function of the number of transmitted bitplanes. Furthermore, it has been shown in [6] that this function does not change its convexity more than once. Hence, a quadratic function is a good approximation for it.

Combining this simplified D-Q function with the $R(z)$ function in (5) yields what we call the semi-analytic square root (sSqrt) R-D model:

$$PSNR(R) = c_1 R + c_2 \sqrt{R} + c_3, \tag{17}$$

where c_1, c_2 need to be estimated from empirical R-D samples, and $c_3 = 10\log_{10}(255^2/\sigma^2)$ for a source with variance σ^2. To implement this model, we determine the bitrate and PSNR values at all bitplane boundaries. Then, we do curve fitting to obtain c_1, c_2, c_3.

D. Semi-analytic Logarithm Model

This model is based on the analytic logarithm model in Section IV-B. However, instead of using parameters of the Laplacian mixture density function, this model employs curve fitting to compute a_1, a_2, a_3, a_4 in (11). This leads to what we call the semi-analytic logarithm (sLog) R-D model. To implement this model, we determine the bitrate and distortion values at all bitplane boundaries. Then we do curve fitting to get a_1, a_2, a_3, a_4.

E. Semi-analytic Generalized Gaussian Function Model

This model is based on the analytic GGF model in Section IV-C. However, instead of numerically computing the integration in (14), this model utilizes a *heuristic* function [5]:

$$PSNR(R) = g_1 R + g_2 - \frac{g_2 - D_b}{1 + g_3 R}, \tag{18}$$

TABLE II

LIST OF TEST VIDEO SEQUENCES.

Class	Sequences	Resolution	# of Frames
Low	*Akiyo*, *Mother*, *Foreman*, Paris, Bridge-close, Bridge-far, Hall-monitor, Highway, Tempete	352x288	300
	Claire, Salesman, Grandma, Carphone	176x144	300
Medium	*Bus*	352x288	150
	Mobile, Container	352x288	300
High	Coastguard	352x288	300
	Garden	352x240	115
	Tennis	352x240	112
	Football	352x240	125

where PSNR is the quality, D_b is the base layer only quality, and g_1, g_2, g_3 are parameters that need to be estimated. This function was proposed after empirically observing the actual R-D data and the R-D estimations produced by the analytic GGF model. To implement this model, we determine the bitrate and PSNR values at all bitplane boundaries. Then, we do curve fitting to derive g_1, g_2, g_3.

VI. Evaluation of Rate-Distortion Models

In this section, we present an extensive experimental study to evaluate the performance of the R-D models described in the previous sections. We first describe the video sequences used in the experiments and why we chose them. Then, we present our experimental setup and the performance metrics considered. Then, we present the results of comparing analytic models, followed by results of comparing semi-analytic models.

A. Selection of Test Video Sequences

Choosing a representative set of video sequences is a critical step in evaluating and analyzing the performance of R-D models. A homogeneous set of sequences may produce biased comparison results, because some models may perform exceptionally well under certain sequences.

We use two key features to characterize video sequences: spatial complexity and temporal complexity. To quantify these complexities, we adopt the neighborhood difference metric [31]. This metric captures the amount of variations between a block and its neighbor by accumulating the differences among corresponding transform coefficients in the two blocks. The neighborhood metric approaches 0 when neighboring blocks are similar, and increases as the similarity between the two blocks decreases. The spatial complexity is measured by averaging all neighborhood differences in the same frame. Whereas the temporal complexity is measured by averaging neighborhood differences between adjacent frames

To form a set of test sequences, we considered twenty video sequences from various sources [32], [33]. Table II lists the considered sequences. We then classify sequences based on their spatial and temporal complexities. Sequences that have high spatial and temporal complexities are what we call high-complexity sequences. In contrast, sequences with low spatial and temporal complexities are called low-complexity sequences. Sequences that have either high temporal or spatial complexity, but not both, are called medium-complexity sequences.

Out of these twenty sequences, we chose seven representative sequences with different complexities: Akiyo, Mother, Foreman, Mobile, Bus, Garden, and Football. We briefly describe these seven sequences. In Akiyo, a female reporter reads news with very limited head movements in front of a fixed camera. There are more movements in Mother, especially the hand and hair movements when compared to Akiyo. The camera is fixed in Mother without any pan, zoom, or movement. Foreman also features a talking person, but it was taken with a hand-held camera that introduces camera movements. Mobile contains saturated colors, thus has higher spatial complexity. Bus was shot with a moving camera following a running bus in short distance, which results in a high temporal complexity. Garden contains intensive colors and was filmed on a moving car, while Football involves lots of complicated movements and details. We believe that these seven test sequences form a diverse enough set to examine the performance of R-D models.

B. Experimental Setup and Performance Metrics

1) Software used and implemented: We use the MPEG-4 Reference Software Version 2.5 [34] developed by Microsoft as an experimental package for the MPEG-4 standard. It is implemented in C++ and contains three major executables: *encoder*, *decoder*, and *fgs_server*. The *encoder* is a configurable MPEG-4 encoder that can compress a raw video file into two bitstreams: base layer and enhancement layer. Each bitstream is stored in a separate file. The *decoder* is FGS-enabled, that is, it can process an incomplete enhancement layer and produces a raw video file with proportional quality improvements. The *fgs_server* is a utility to trim the enhancement layer according to a given target bitrate. It does so by calculating the number of allowed bits in each frame at the target bitrate. These bits, which represent

```
1.  Choose a sequence from the test sequences (described in Section VI-A);
2.  FGS-encode the sequence with base layer rates $R_b$ at $8$, $16$, $32$, $64$, and $128$ Kbps;
3.  **for** each R-D model $model \in \{Sqrt, Log, GGF, Linear, sSqrt, sLog, sGGF\}$ **do**
4.    **for** each base layer rate $R_b$ **do**
5.      **for** each frame $f$ in the sequence **do**
6.        Construct the set $\mathbb{R}_s$ of sampling bitrates by choosing equally-spaced
          $K$ samples from each bitplane;
7.        **for** each sampling rate $R_s \in \mathbb{R}_s$ **do**
8.          Measure actual distortion $D^{Emp}(R_s)$ by comparing reconstructed and
            original frames;
9.          Estimate distortion using the R-D model $D^{model}(R_s)$;
10.         Compute the distortion deviation as: $\varepsilon_f(R_s) = |D^{emp}(R_s) - D^{model}(R_s)|$;
11.       **endfor**
12.       Compute the average distortion deviation per frame;
13.       Maintain the average distortion deviation for each frame type: I, P, B;
14.     **endfor**
15.     Compute the average and maximum distortion deviations across all frames;
16.     Measure the total running time;
17.     Compute the normalized range of applicability;
18.   **endfor**
19. **endfor**
20. Repeat steps 1—19 for another video sequence;
```

Fig. 4. Our procedure for rigorously evaluating the performance of various R-D models.

the truncated bitstream, are then saved in a new file. We have instrumented the reference software to extract various characteristics of an input video sequence. For instance, we collect transform coefficients, number of bitplanes, and size of each bitplane in the enhancement layer. This information is used to estimate parameters of the R-D models.

We have implemented all R-D models described in this paper, a total of eight models. Specifically, we have implemented the three analytic R-D models: square root (denoted by Sqrt in the plots), logarithm (Log), and generalized Gaussian (GGF); the pure empirical model (Emp), the piecewise linear (Linear), and the three semi-analytic R-D models: semi-analytic square root (sSqrt), semi-analytic logarithm (sLog), and semi-analytic generalized Gaussian (sGGF). All models are implemented in Matlab.

2) Selection of Sampling bitrates: We evaluate the R-D models across a wide range of bitrates, not only at bitplane boundaries. This is important because streaming applications using FGS-encoded sequences are allowed to truncate bitstreams at arbitrary *bit* positions. It is also critical to emphasize on the range of bitrates that most applications will heavily use. To achieve these two goals, we propose the following sampling scheme. Suppose Z is the total number of bitplanes, l_z is the size of bitplane z ($z = 1, 2, \ldots, Z$), and T is the frame period. We choose K equally-spaced bitrates in each bitplane. Specifically, we form the set of sampling bitrates $\mathbb{R}_s$ as

$$\mathbb{R}_s = \{0\} \cup \{R_s^{z,k} \mid z = 1, 2, \ldots, Z \text{ and } k = 1, 2, \ldots, K\}, \tag{19}$$

where $R_s^{z,k} = \frac{\sum_{u=1}^{z-1} l_u + (l_z/K)(k)}{T}$.

The set $\mathbb{R}_s$ achieves the above two goals, because it covers the whole bitrate range from base layer only ($R_s = 0$) to full quality ($R_s = R_s^{Z,K}$). Furthermore, since the most significant bitplanes are typically small in size, gaps between sampling bitrates will be small. Therefore, the accuracy of the considered R-D model will be higher in these significant bitplanes. The R-D performance of significant bitplanes is crucial for two reasons. First, significant bitplanes have high per-bit reduction rates in distortion. This implies more sampling bitrates are needed to capture the rapid distortion variations. Second, significant bitplanes are streamed more often owing to the embedded property of FGS-encoded streams. The embedded property states that a lower-rate bitstream is always a prefix of a higher-rate one.

3) Performance Metrics: We consider the following performance metrics for each R-D model: accuracy, range of applicability, and time complexity.

We consider model accuracy at different granularities: for individual sampling bitrates, for every frame, and for the whole sequence. We measure accuracy as the difference between the quality (in PSNR) predicted by the considered R-D model and the actual quality at a given bitrate. Specifically, for every sampling bitrate $R_s \in \mathbb{R}_s$, we compute the absolute distortion deviation (or error) as:

$$\varepsilon_f(R_s) = |D^{emp}(R_s) - D^{model}(R_s)|, \tag{20}$$

where $D^{emp}(R_s)$ is the actual distortion computed by comparing the reconstructed and original frames, and $D^{model}(R_s)$ is the distortion estimated by the R-D model for frame f at bitrate R_s. For the frame level, we compute the average distortion deviation over all sampling bitrates. We also compute the average distortion deviation for different frame types (I, P, and B). For the sequence level, we compute the average and maximum deviations across all frames.

Ideally, an R-D model should provide valid—finite and nonnegative—distortion estimations for all sampling bitrates R_s. Unfortunately, some R-D models only support a subset of the whole bitrate range. To capture this important aspect, we use the range of applicability metric. The range of applicability of an R-D model is delimited by the smallest bitrate beyond which that model fails to provide valid results. To enable cross-sequence comparisons, we normalize the range of applicability by dividing it by the maximum bitrate of the sequence. The maximum bitrate of the sequence occurs when all bitplanes are transmitted.

Time complexity is also an important metric especially for streaming applications that use R-D models in real-time, e.g., video conferencing. We measure the running time to construct R-D models. To mitigate the effect of clock precisions and interruptions from the operating system, we measure the running time for all frames in a video sequence and report the average per-frame running time. All experiments are conducted on a 2.8 GHz Pentium 4 workstation.

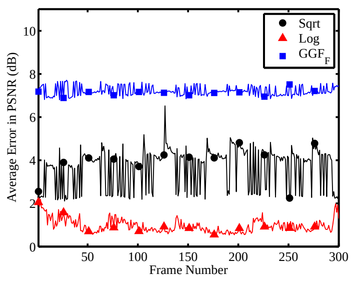
4) Evaluation Procedure: We evaluate the R-D models on the set of video sequences described in Section VI-A. We FGS-encode every video sequence at several different bitrates for the base layer. The rate of the base layer directly impacts the number and size of bitplanes in the enhancement layer. Therefore, we are effectively evaluating the models under different relative sizes of the base and enhancement layers. This is important because streaming systems work in different network environments, and they typically adjust the base layer rate based on these conditions. In addition, for every video sequence encoded at a given base layer rate, we decode the enhancement layer at different rates. This is done by truncating the bitstream at the appropriate bit positions. Decoding at different rates captures the heterogeneous nature of clients and the diverse communication channels over which they receive video sequences. Figure VI-A summarizes the procedure that we follow in evaluating the performance of R-D models.

We believe that our experimental set up and evaluation procedure rigorously evaluate the R-D models, because they account for various source characteristics, diverse communication channel conditions, and different encoding/decoding parameters.

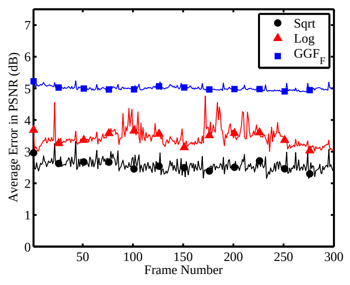
C. Results for Analytic Models

We first discuss accuracy of the analytic R-D models, followed by their applicable ranges and time complexities.

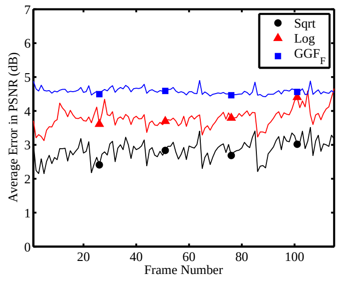
1) Accuracy: We plot the frame-level average distortion deviation in Figure 5. We present results of a sample sequence in each complexity class (from left to right: low, medium, and high complexity) in each vertical column of subfigures. Each horizontal row represents a different base layer rate R_b. We



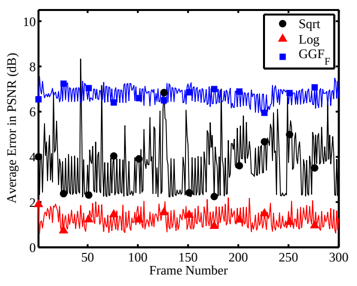
(a) Akiyo $R_b = 8$ Kbps



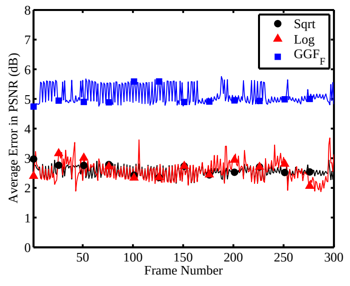
(b) Mobile $R_b = 8$ Kbps



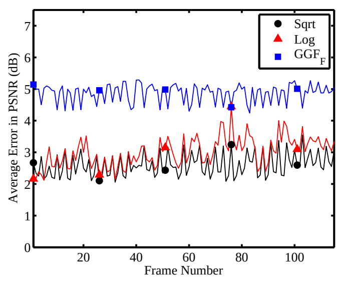
(c) Garden $R_b = 8$ Kbps



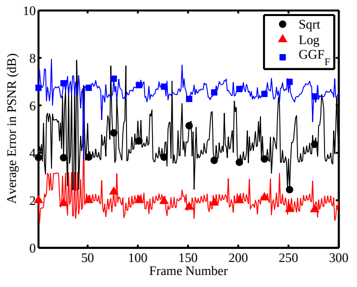
(d) Akiyo $R_b = 32$ Kbps



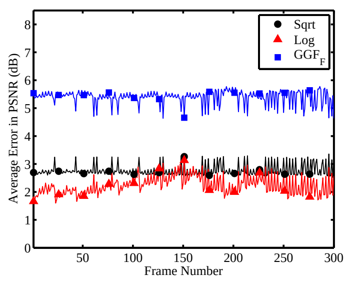
(e) Mobile $R_b = 32$ Kbps



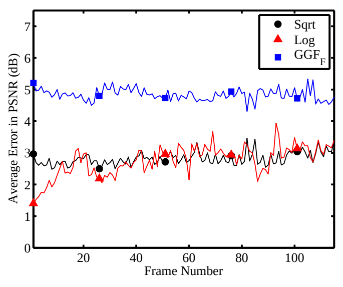
(f) Garden $R_b = 32$ Kbps



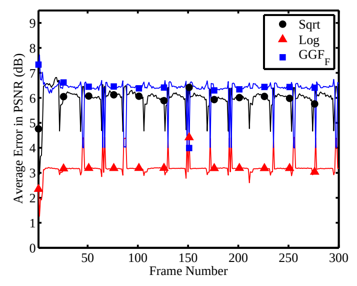
(g) Akiyo $R_b = 64$ Kbps



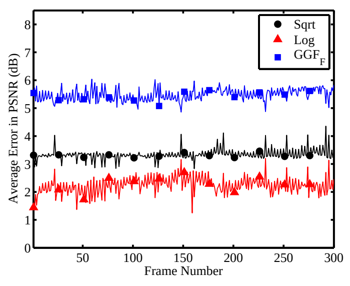
(h) Mobile $R_b = 64$ Kbps



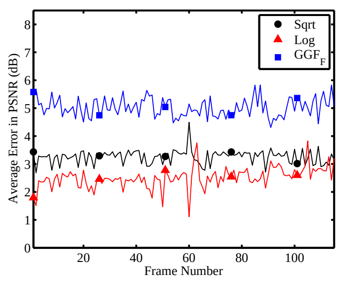
(i) Garden $R_b = 64$ Kbps



(j) Akiyo $R_b = 128$ Kbps



(k) Mobile $R_b = 128$ Kbps



(l) Garden $R_b = 128$ Kbps

Fig. 5. The frame-level average distortion deviation of the R-D models of three video sequences of different complexities, where the base layers are coded with different bitrates.

make a few observations on this figure. First, the GGF model produces the highest distortion deviation, at least 4 dB, in all cases. Second, the logarithm model performs better than the square root model in low complexity sequences. Furthermore, in medium and high complexity sequences, the logarithm model still outperforms the square root model for base layer bitrate $R_b > 32$ Kbps. These two models have comparable accuracy for $R_b = 32$ Kbps; the square root model performance is improved for $R_b < 32$ Kbps.

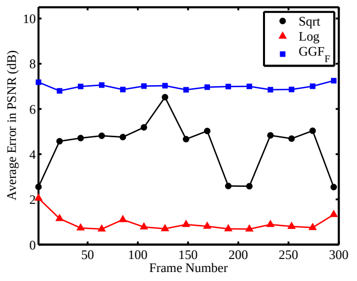
We also study the accuracy for different frame types. Plots of the average distortion deviation for I-, P-, and B-frames are given in Figures 6, 7, and 8, respectively. The results show the same relative accuracy. This indicates the frame type does not affect relative accuracy of these models.

We present the sequence-level average and maximum distortion deviation for different sequences coded at different base layer rates R_b in Figures 9–12, where the sequences are sorted in ascending order of their complexities. As shown in Figure 11, the logarithm model is the most accurate in terms of the average distortion deviation with at most 3 dB deviation, while the square root model produces as high as 6 dB and the GGF model results in as high as 7 dB. Comparing the square root model versus the logarithm model, we observe that: (i) The square root model results in higher distortion deviations for lower complexity sequences as shown in Figures 11(b), 11(c), 12(a), and 12(b). (ii) The logarithm model produces higher distortion deviations for higher complexity sequences that are coded with low base layer rates R_b as shown in in Figures 11(a) and 12(a).

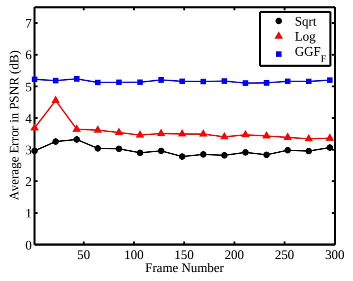
Then, we study the shape of R-D curves for a few frames. Figure 13 illustrates that: (i) The GGF model suffers high distortion deviations at low bitrates, as the high resolution hypothesis does not hold when the quantization step is large; and (ii) The logarithm and square root models over estimate the quality at high bitrates. More importantly, the logarithm model does not produce a non-decreasing R-D curve in some cases, a sample result is shown in Figure 13(c). This is inaccurate because higher bitrates should always lead to better quality. We further investigate the root cause of the problem.

The logarithm model utilizes a linear R-Q model which assumes that the percentage of non-zero quantized coefficients ζ is a linear function of the bitrate. Through extensive experiments, we find that this linearity is strong in the more significant bitplanes, especially in the first four bitplanes. Since low complexity sequences only have a few bitplane, the linear relationship is strong for these sequences. In contrast, for high complexity sequences coded at low base layer rates, this linearity starts to break. We present the actual ζ-R curve and the estimated one in Figure 14. This figure explains the abnormal R-D curves at high bitrates observed in Figures 13(a)–13(c).

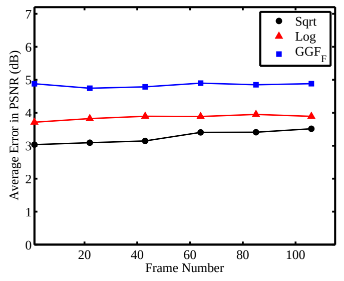
In summary, the GGF model is the least accurate in all cases, while the logarithm model is the most



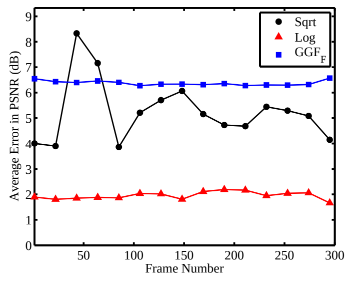
(a) Akiyo $R_b = 8$ Kbps



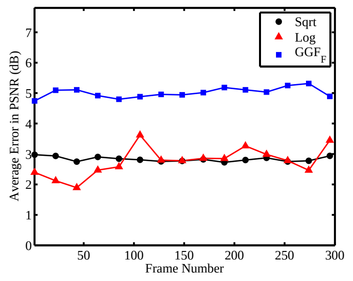
(b) Mobile $R_b = 8$ Kbps



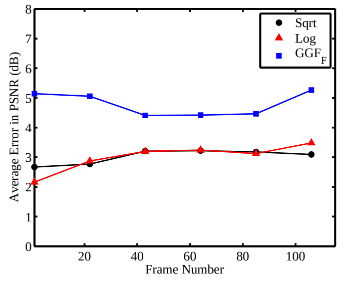
(c) Garden $R_b = 8$ Kbps



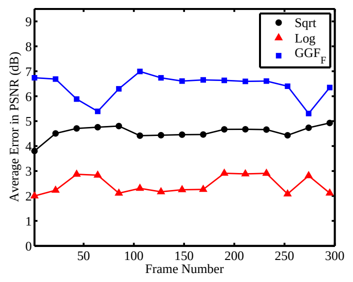
(d) Akiyo $R_b = 32$ Kbps



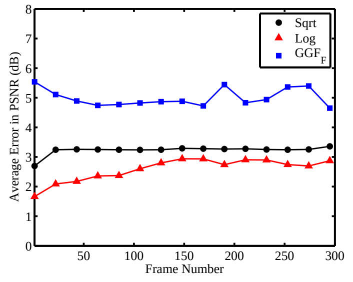
(e) Mobile $R_b = 32$ Kbps



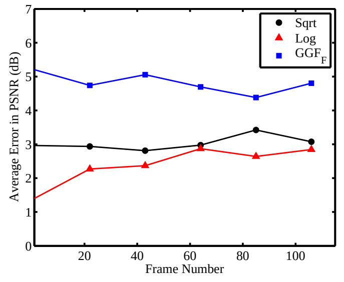
(f) Garden $R_b = 32$ Kbps



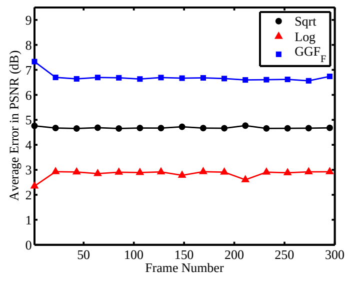
(g) Akiyo $R_b = 64$ Kbps



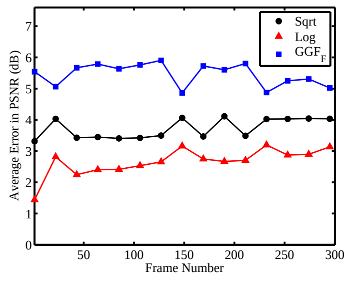
(h) Mobile $R_b = 64$ Kbps



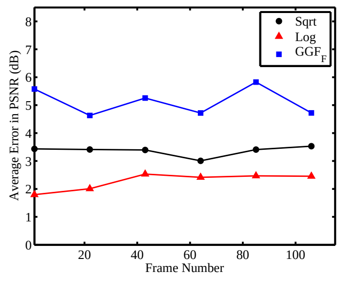
(i) Garden $R_b = 64$ Kbps



(j) Akiyo $R_b = 128$ Kbps

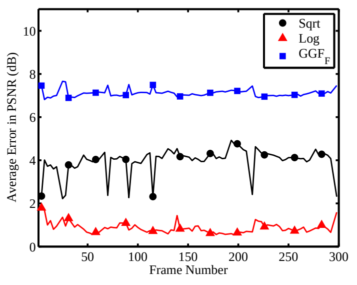


(k) Mobile $R_b = 128$ Kbps

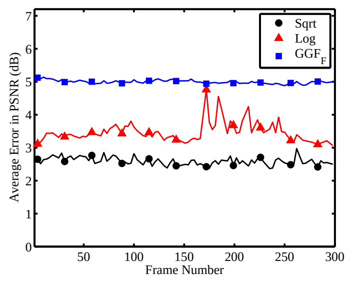


(l) Garden $R_b = 128$ Kbps

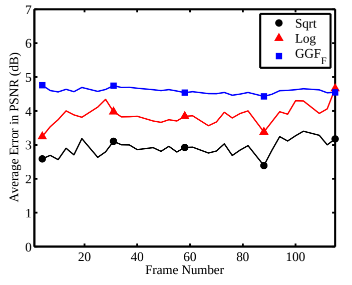
Fig. 6. The frame-level average distortion deviation of the R-D models of three video sequences of different complexities, where the base layers are coded with different bitrates. Only I-frame are considered.



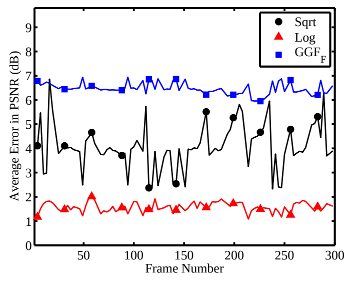
(a) Akiyo $R_b = 8$ Kbps



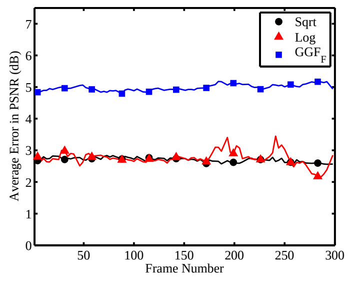
(b) Mobile $R_b = 8$ Kbps



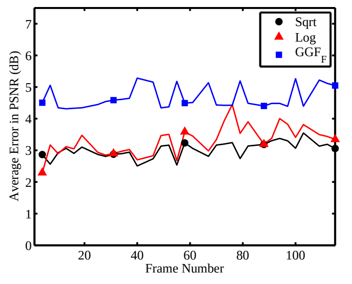
(c) Garden $R_b = 8$ Kbps



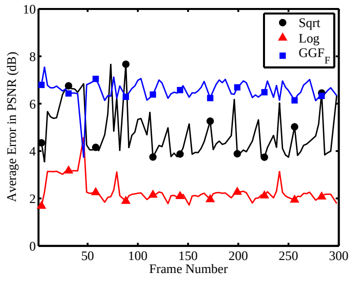
(d) Akiyo $R_b = 32$ Kbps



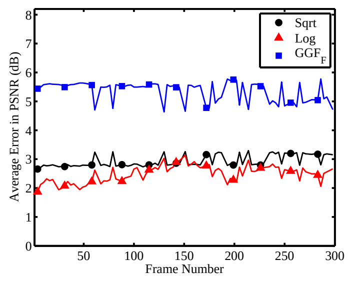
(e) Mobile $R_b = 32$ Kbps



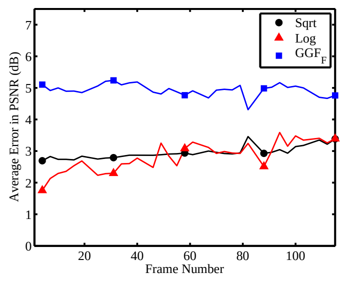
(f) Garden $R_b = 32$ Kbps



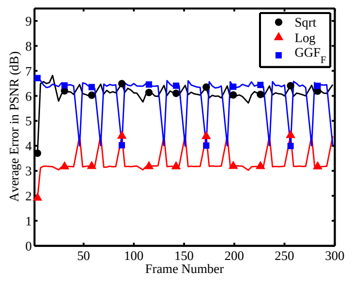
(g) Akiyo $R_b = 64$ Kbps



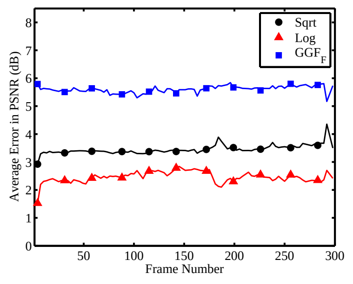
(h) Mobile $R_b = 64$ Kbps



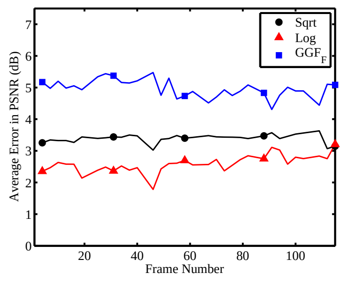
(i) Garden $R_b = 64$ Kbps



(j) Akiyo $R_b = 128$ Kbps

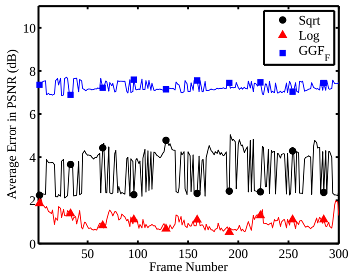


(k) Mobile $R_b = 128$ Kbps

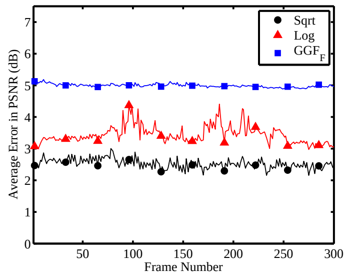


(l) Garden $R_b = 128$ Kbps

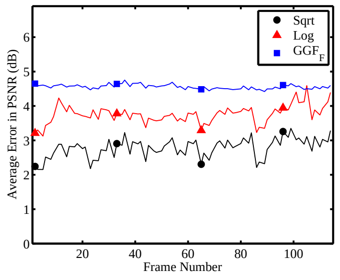
Fig. 7. The frame-level average distortion deviation of the R-D models of three video sequences of different complexities, where the base layers are coded with different bitrates. Only P-frames are considered.



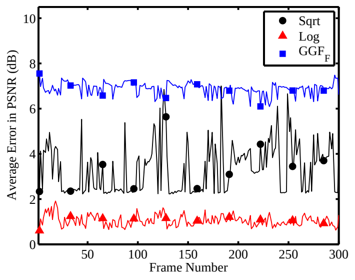
(a) Akiyo $R_b = 8$ Kbps



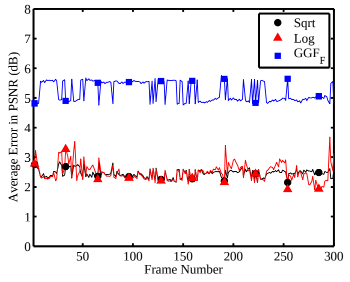
(b) Mobile $R_b = 8$ Kbps



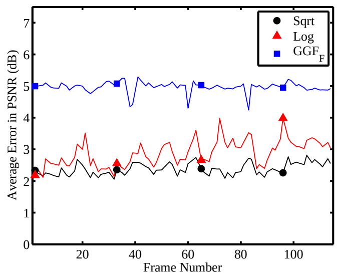
(c) Garden $R_b = 8$ Kbps



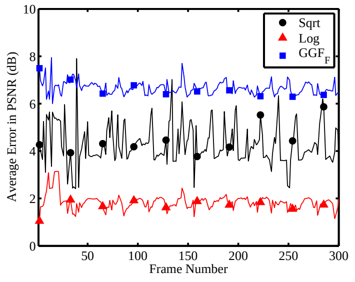
(d) Akiyo $R_b = 32$ Kbps



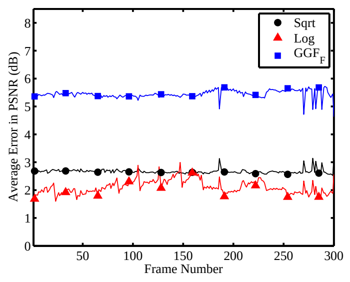
(e) Mobile $R_b = 32$ Kbps



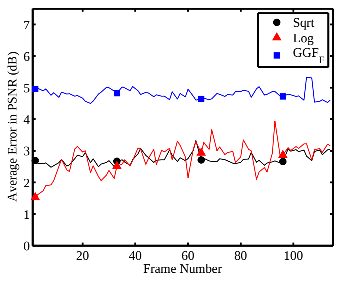
(f) Garden $R_b = 32$ Kbps



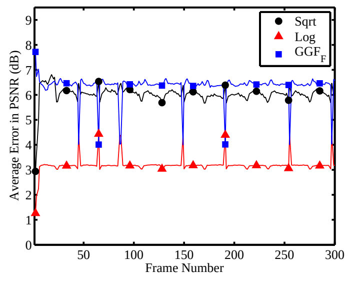
(g) Akiyo $R_b = 64$ Kbps



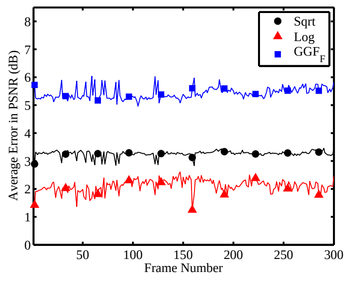
(h) Mobile $R_b = 64$ Kbps



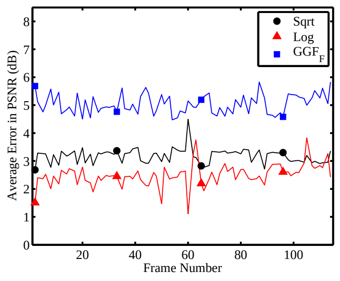
(i) Garden $R_b = 64$ Kbps



(j) Akiyo $R_b = 128$ Kbps

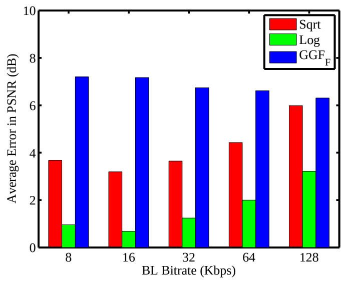


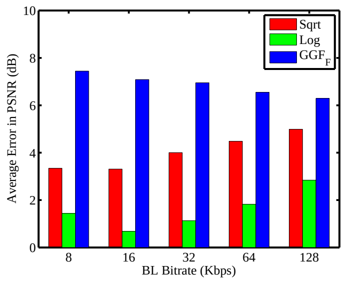
(k) Mobile $R_b = 128$ Kbps

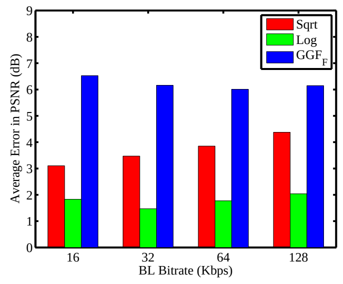


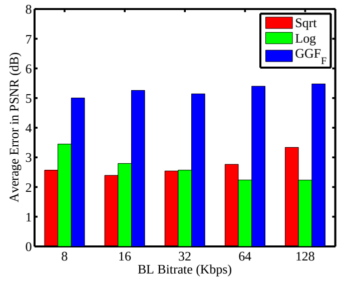
(l) Garden $R_b = 128$ Kbps

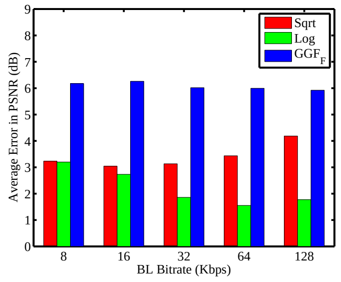
Fig. 8. The frame-level average distortion deviation of the R-D models of three video sequences of different complexities, where the base layers are coded with different bitrates. Only B-frames are considered.


(a) Akiyo


(b) Mother


(c) Foreman


(d) Mobile


(e) Bus

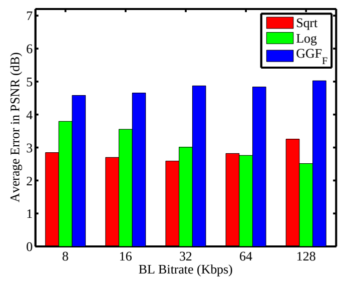
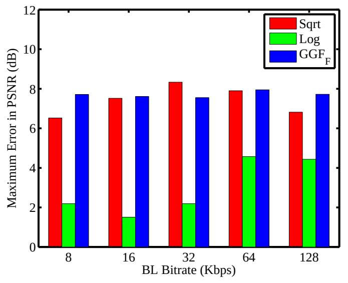
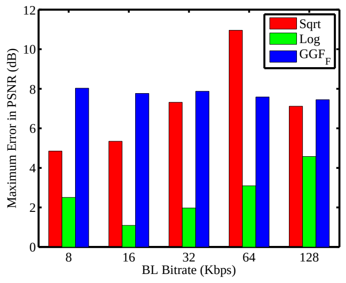
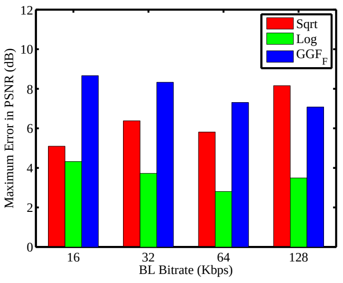
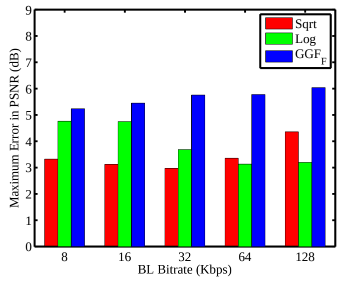

(f) Garden

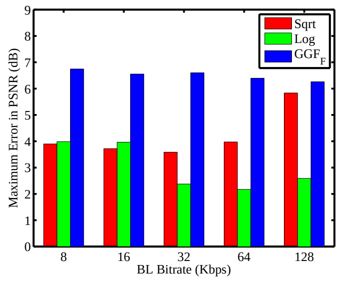
Fig. 9. The average error of the R-D models across all frames of six video sequences coded at different base layer bitrates.


(a) Akiyo


(b) Mother


(c) Foreman


(d) Mobile


(e) Bus

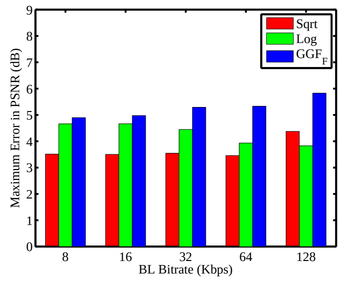
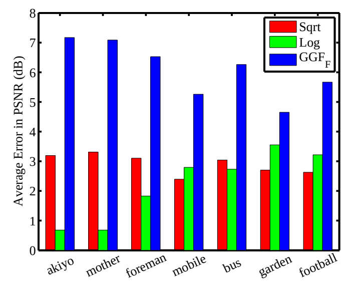
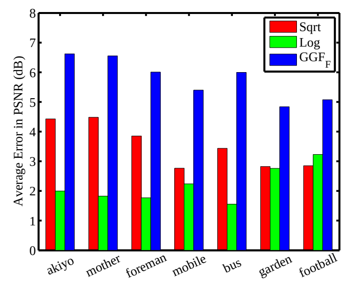

(f) Garden

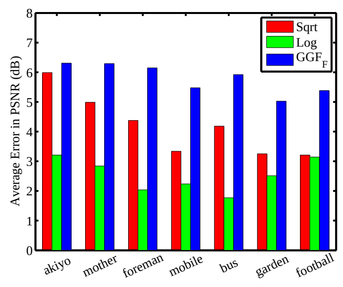
Fig. 10. The maximum error of the R-D models across all frames of six video sequences coded at different base layer bitrates.



(a) $R_b = 16$ Kbps

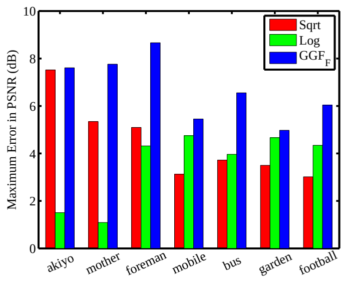


(b) $R_b = 64$ Kbps

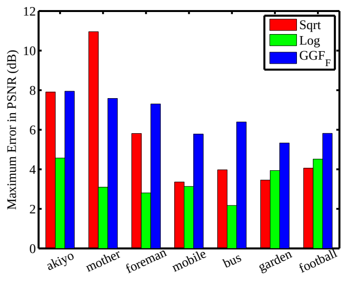


(c) $R_b = 128$ Kbps

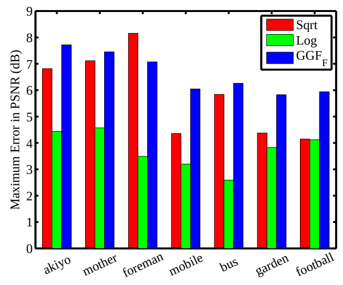
Fig. 11. The average error of the R-D models across all frames of all video sequences of different complexities.



(a) $R_b = 16$ Kbps



(b) $R_b = 64$ Kbps

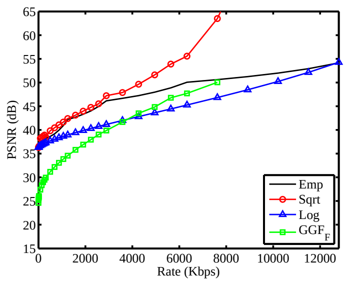


(c) $R_b = 128$ Kbps

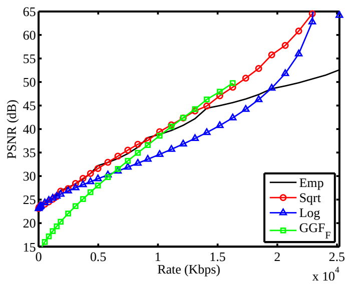
Fig. 12. The maximum error of the R-D models across all frames of all video sequences of different complexities.

accurate except for medium or high complexity sequences that are coded at base layer rates less than 32 Kbps. The square root model is not accurate for low complexity sequences coded at high base layer bitrates, because these coded streams: (i) have source distributions with large variation rates that contradict the high frequency hypothesis, and (ii) have a few bitplanes, therefore, the estimation inaccuracy in the least significant bitplanes dominates the overall performance. The performance of the logarithm model suffers in high complexity sequences coded at low base layer bitrates, because the R-Q relationship of these code streams deviates from the assumed linear R-Q model.

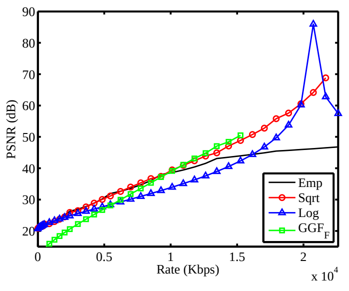
2) Range of Applicability: We present the range of applicability in Figure 15, which shows that: (i) The logarithm model supports all bitrates—from 0 to the full quality; (ii) The square root model is less applicable in the least significant bitplanes: since low complexity sequences have a few bitplanes, the square root model is applicable only for about 70% of bitrates in these sequences; however, it is applicable for almost 95% of bitrates in medium and high complexity sequences; and (iii) The GGF



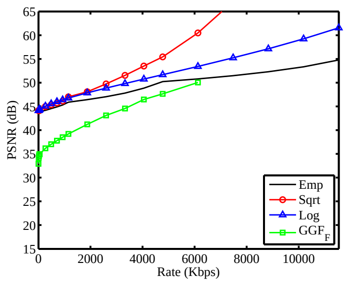
(a) Akiyo frame 1, $R_b = 8$ Kbps



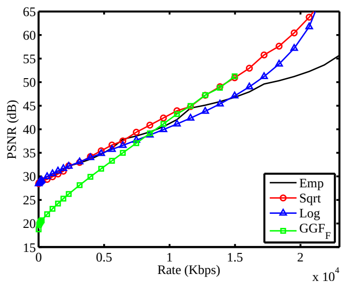
(b) Mobile frame 171, $R_b = 8$ Kbps



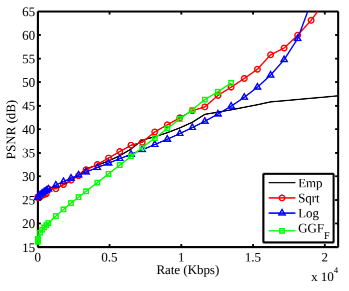
(c) Garden frame 100, $R_b = 8$ Kbps



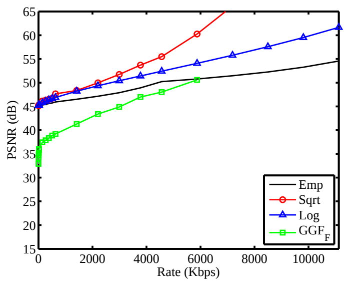
(d) Akiyo frame 1, $R_b = 32$ Kbps



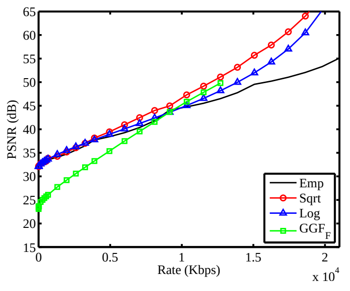
(e) Mobile frame 171, $R_b = 32$ Kbps



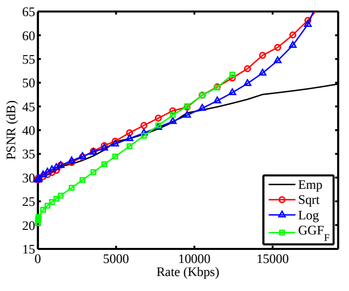
(f) Garden frame 100, $R_b = 32$ Kbps



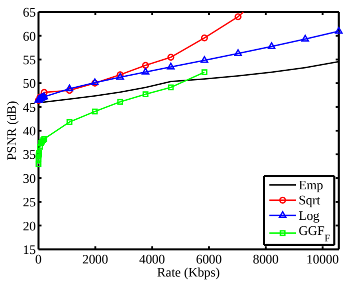
(g) Akiyo frame 1, $R_b = 64$ Kbps



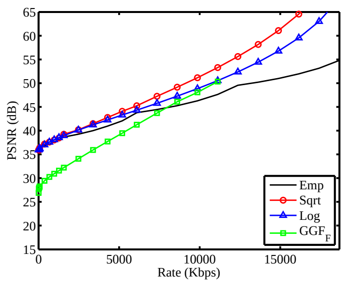
(h) Mobile frame 171, $R_b = 64$ Kbps



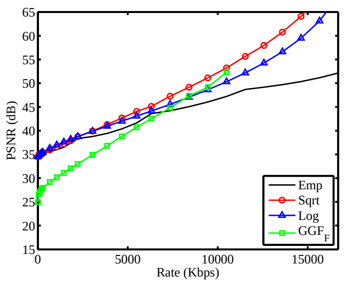
(i) Garden frame 100, $R_b = 64$ Kbps



(j) Akiyo frame 1, $R_b = 128$ Kbps

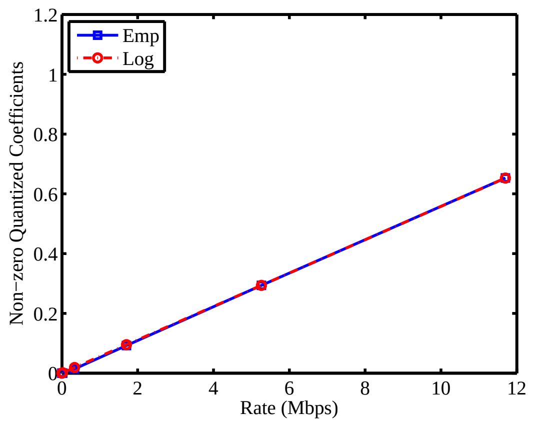


(k) Mobile frame 171, $R_b = 128$ Kbps

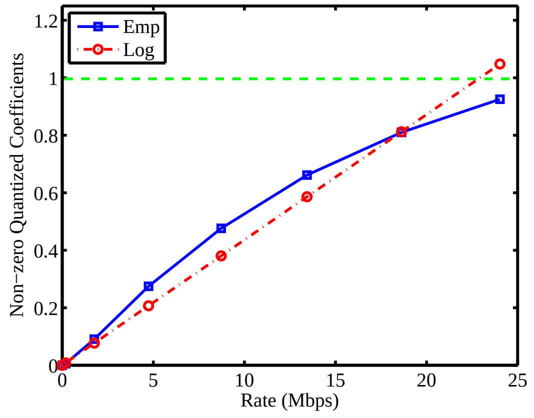


(l) Garden frame 100, $R_b = 128$ Kbps

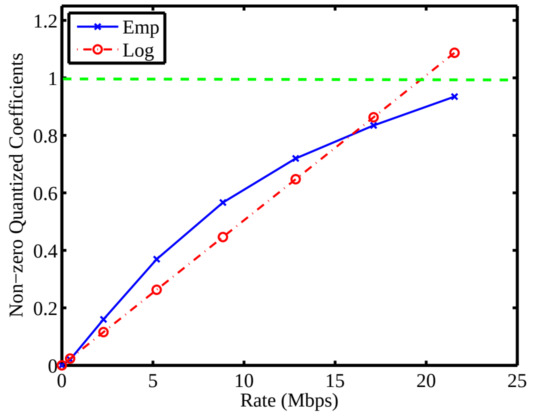
Fig. 13. R-D curves for four models applied to three video sequences of different complexities, where the base layers are coded at different bitrates.



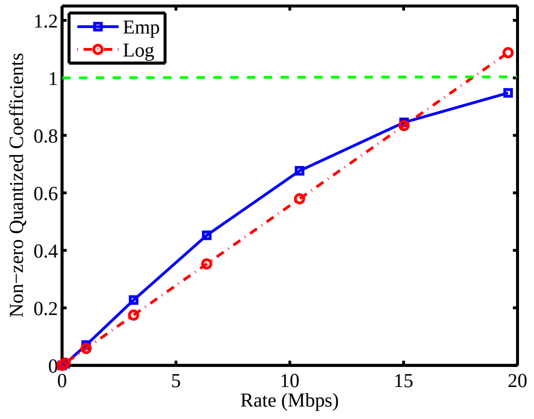
(a) Akiyo sequence, frame 298, $R_b = 16$ Kbps



(b) Mobile sequence, frame 171, $R_b = 8$ Kbps



(c) Garden sequence, frame 100, $R_b = 8$ Kbps

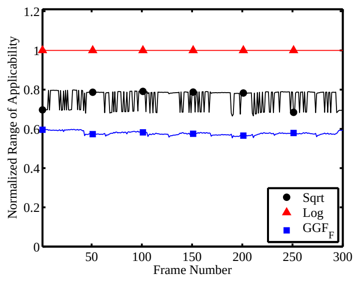


(d) Football sequence, frame 35, $R_b = 16$ Kbps

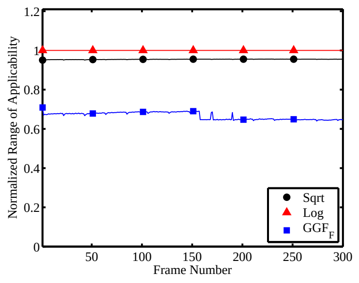
Fig. 14. Although ζ-R shows a strong linearity in low complexity sequences (a); it is imperfect in medium and high complexity sequences (b), (c), and (d). We observed that the ζ-R curve shows strong linearity in the first four bitplanes, but not in other bitplanes. This inaccuracy results in erroneous R-D estimations.

model's applicability is no more than 70% in all cases.

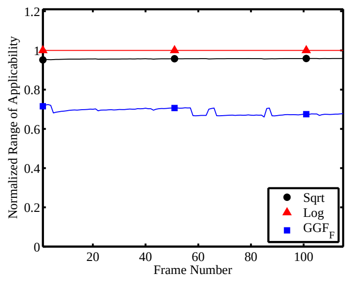
3) Time Complexity: We show the average running time for each frame in Figure 16. This figure illustrates that: (i) The GGF model is the most efficient; and (ii) The square root model is slightly faster than the logarithm model, because the latter requires a time-intensive estimation of its R-Q model parameter γ.



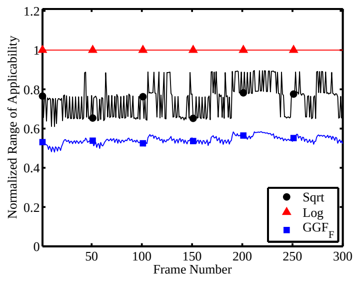
(a) Akiyo $R_b = 8$ Kbps



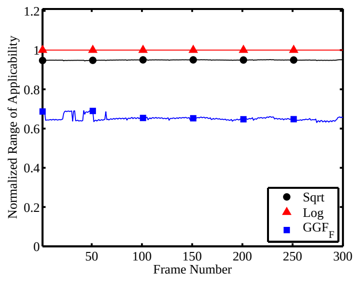
(b) Mobile $R_b = 8$ Kbps



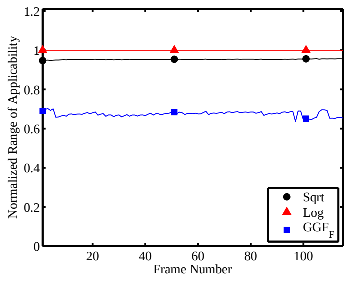
(c) Garden $R_b = 8$ Kbps



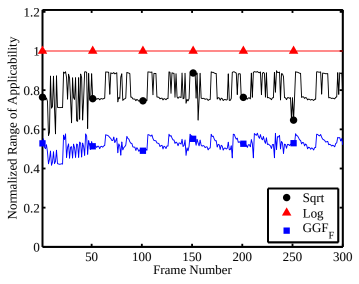
(d) Akiyo $R_b = 32$ Kbps



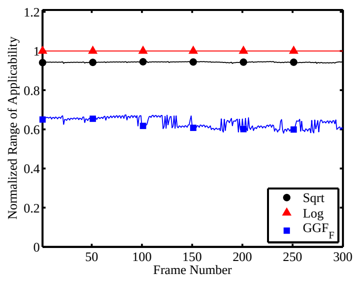
(e) Mobile $R_b = 32$ Kbps



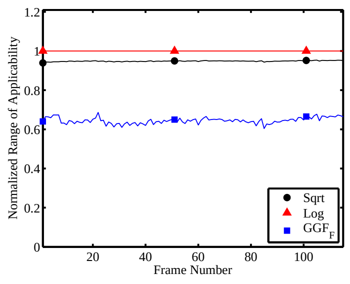
(f) Garden $R_b = 32$ Kbps



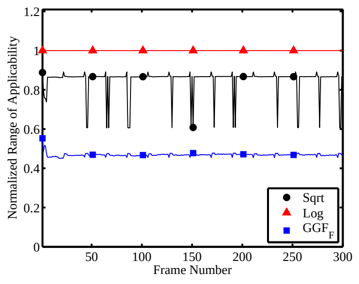
(g) Akiyo $R_b = 64$ Kbps



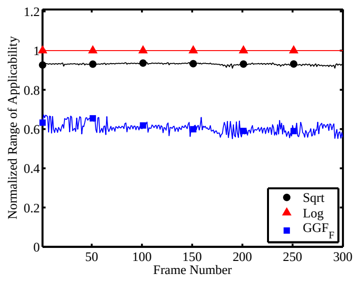
(h) Mobile $R_b = 64$ Kbps



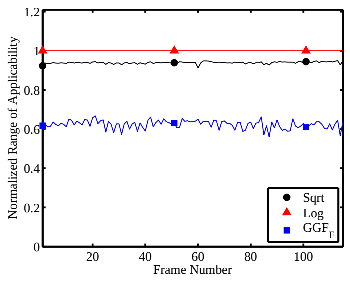
(i) Garden $R_b = 64$ Kbps



(j) Akiyo $R_b = 128$ Kbps

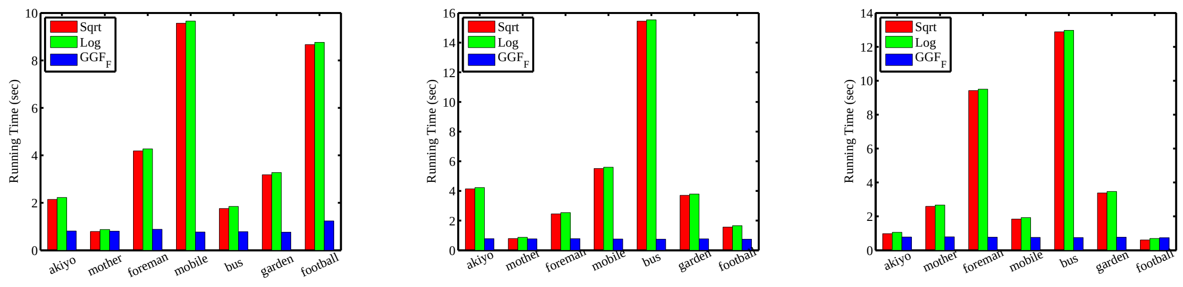


(k) Mobile $R_b = 128$ Kbps



(l) Garden $R_b = 128$ Kbps

Fig. 15. The normalized applicable range of the R-D models across all frames of three video sequences with different complexities, where the base layers are coded with different bitrates.



(a) $R_b = 16$ Kbps (b) $R_b = 64$ Kbps (c) $R_b = 128$ Kbps

Fig. 16. The average running time across all frames of all video sequences of different complexities.

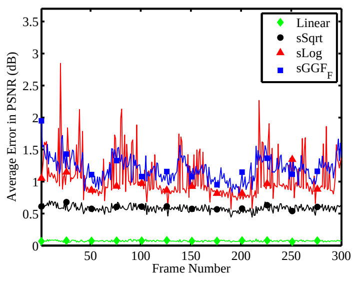
D. Results for Empirical and Semi-analytic Models

We present and analyze the results for empirical and semi-analytic models. We consider one empirical model—the piecewise linear (Linear) model—and three semi-analytic models: the square root (sSqrt), the logarithm (sLog), and the GGF (sGGF$_\text{F}$). We implemented these models using the nonlinear least-squares fitting subroutine provided by Matlab.

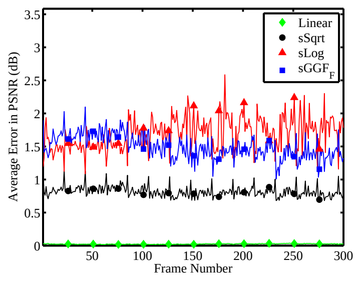
1) Accuracy: Figure 17 shows the frame-level average distortion deviation of these models. We observe that: (i) the piecewise linear model produces negligible, always less than 0.2 dB, error in all cases; (ii) the square root model has up to 1 dB deviation and is more accurate than the GGF and the logarithm model; and (iii) the logarithm model results in the highest estimation error in almost all cases.

We also notice that the semi-analytic models result in more fluctuations of frame-level average error. This is because the non-linear optimization methods may not lead to a global optimal solution, and sometimes they do not even terminate. Therefore, the methods are often associated with a maximum number of iterations. We set the iteration limit to be 1000 times. In contrast, the piecewise linear model is derived using a simple arithmetic—connecting two points with a line. This is not only more robust, but also much more efficient.

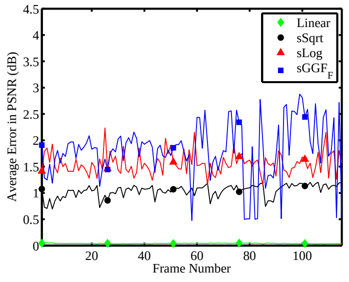
Next, we study the shape of R-D curves in Figure 18. We see abnormal R-D curves caused by the imperfect curve fitting: for instance, the semi-analytic logarithm model produces a decreasing R-D segment in Figure 18(h). In addition, Figure 18 illustrates that while other models produce accurate estimations only at bitplane boundaries; the piecewise linear model results in accurate estimations in the entire bitplane. This explains the highest accuracy provided by the piecewise linear model.



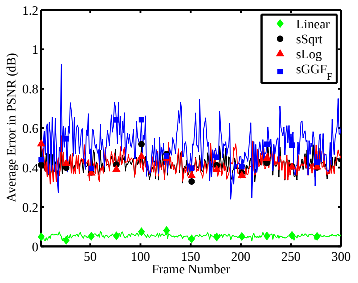
(a) Akiyo $R_b = 8$ Kbps



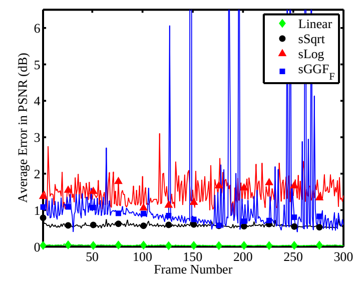
(b) Mobile $R_b = 8$ Kbps



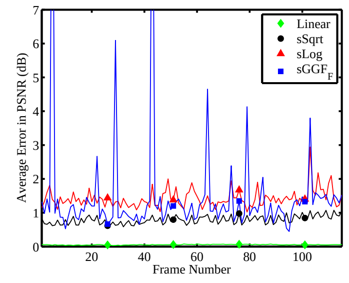
(c) Garden $R_b = 8$ Kbps



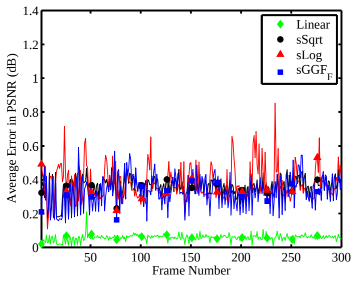
(d) Akiyo $R_b = 32$ Kbps



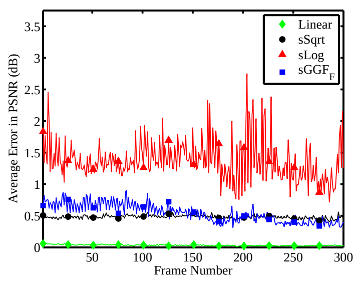
(e) Mobile $R_b = 32$ Kbps



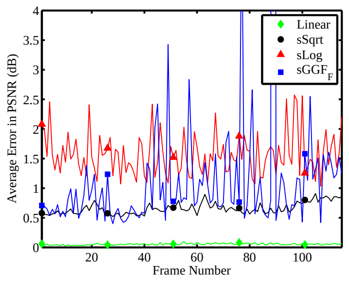
(f) Garden $R_b = 32$ Kbps



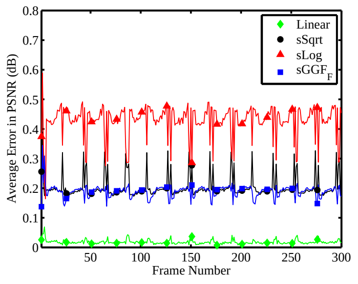
(g) Akiyo $R_b = 64$ Kbps



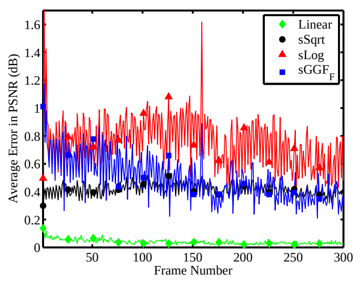
(h) Mobile $R_b = 64$ Kbps



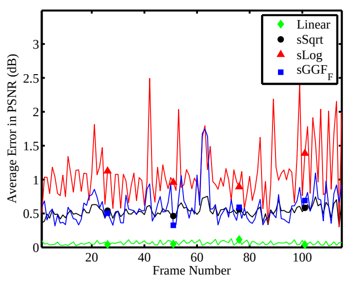
(i) Garden $R_b = 64$ Kbps



(j) Akiyo $R_b = 128$ Kbps

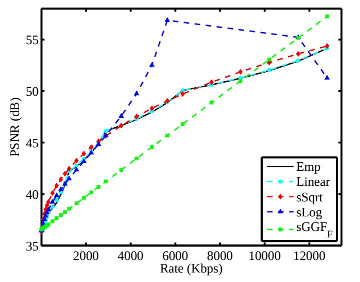


(k) Mobile $R_b = 128$ Kbps

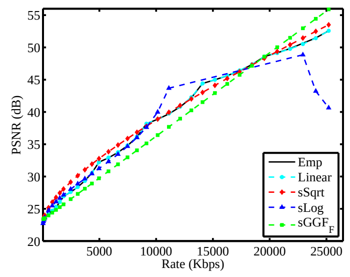


(l) Garden $R_b = 128$ Kbps

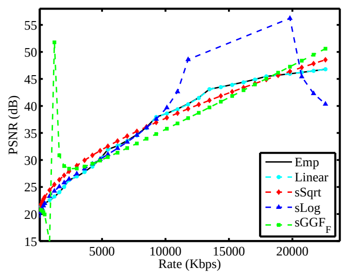
Fig. 17. The average distortion deviation of the empirical and semi-analytic R-D models across all frames of three video sequences of different complexities, where the base layers are coded with different bitrates.



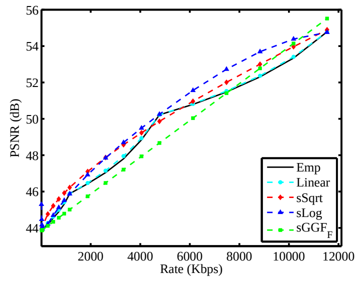
(a) Akiyo frame 1, $R_b = 8$ Kbps



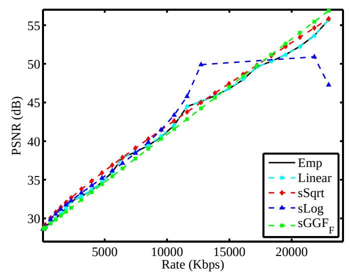
(b) Mobile frame 171, $R_b = 8$ Kbps



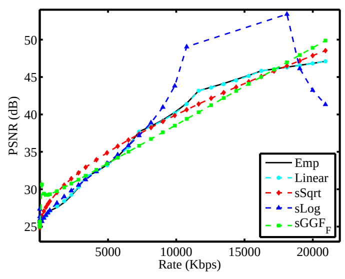
(c) Garden frame 100, $R_b = 8$ Kbps



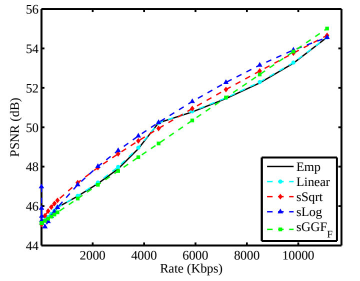
(d) Akiyo frame 1, $R_b = 32$ Kbps



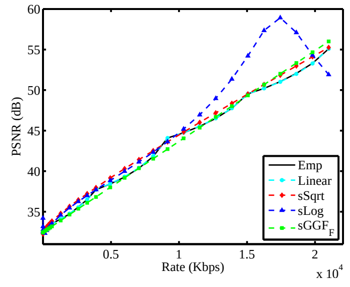
(e) Mobile frame 171, $R_b = 32$ Kbps



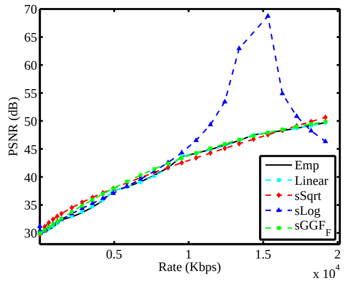
(f) Garden frame 100, $R_b = 32$ Kbps



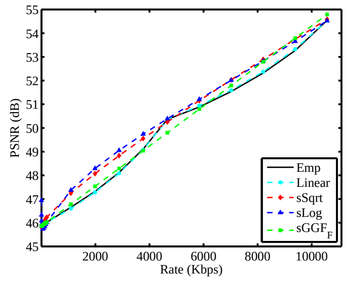
(g) Akiyo frame 1, $R_b = 64$ Kbps



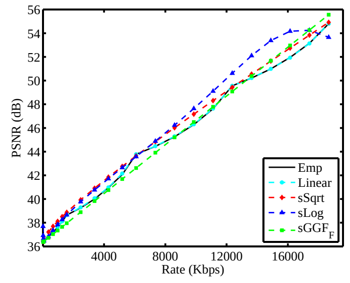
(h) Mobile frame 171, $R_b = 64$ Kbps



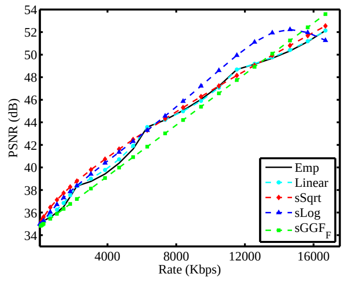
(i) Garden frame 100, $R_b = 64$ Kbps



(j) Akiyo frame 1, $R_b = 128$ Kbps

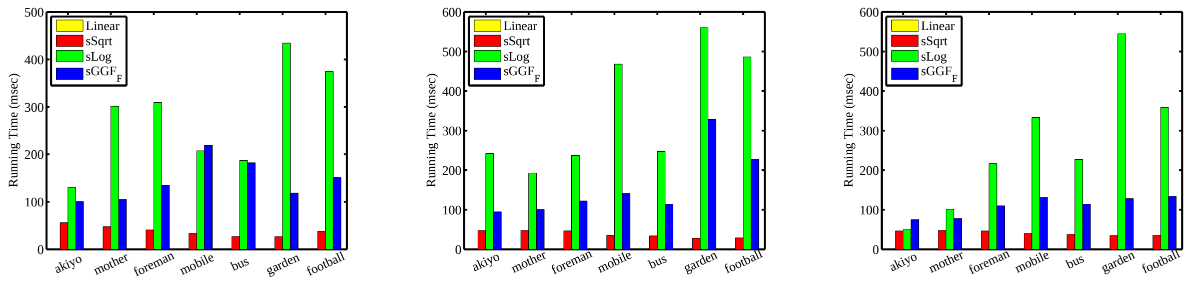


(k) Mobile frame 171, $R_b = 128$ Kbps



(l) Garden frame 100, $R_b = 128$ Kbps

Fig. 18. R-D curves for five empirical and semi-analytic models applied to three video sequences of different complexities, where the base layers are coded at different bitrates.



(a) $R_b = 16$ Kbps (b) $R_b = 64$ Kbps (c) $R_b = 128$ Kbps

Fig. 19. The average running time of the empirical and semi-analytic R-D models across all frames of all video sequences of different complexities.

2) Time Complexity: Figure 19 implies that while the piecewise linear model has negligible time complexity, the semi-analytic models require at least tens of milliseconds to terminate. We find that the logarithm model requires the longest running time, up to 600 milliseconds in the worst case, because of its high non-linearity. The square root is the most efficient semi-analytic model.

VII. Conclusions and Recommendations for Choosing R-D Models

A. Conclusions

We investigated various rate-distortion (R-D) models for fine-grained scalable video sequences. We classified R-D models into three categories: analytic, empirical, and analytically-inspired empirical (or semi-analytic). We have analyzed the three analytical R-D models known in the literature: square-root (Sqrt), logarithm (Log), and generalized Gaussian function (GGF). We also presented three semi-analytic models based on the above analytic models: sSqrt, sLog, and sGGF. In addition, we discussed how a pure empirical model can be implemented by measuring the distortion at equally-spaced sampling bitrates. Finally, we presented the piecewise linear model, which improves upon the pure empirical model by choosing sampling bitrates at bitplane boundaries and connecting these samples by line segments.

We presented systematic ways (pseudo codes) for constructing the R-D models from a given video sequence. We implemented all of the above eight R-D models in Matlab, and we evaluated them using a large and diverse set of carefully-chosen video sequences. In our evaluation process, we considered various source characteristics, diverse channel conditions, different encoding/decoding parameters, different frame types, and several performance metrics including accuracy, range of applicability, and time complexity of each model.

The findings from our extensive experimental study can be summarized as follows:

1) The GGF model shows large deviations from actual distortions. It also does not produce valid results at high bitrates. Therefore, we believe it will be of little use in practice.
2) If the base layer is encoded at high bitrate (> 32 Kbps), the logarithm model produces the most accurate distortion estimations.
3) If the base layer is encoded at lower bitrates (< 32 Kbps) *and* the video sequence has high or medium complexity, the square root model is more accurate than the logarithm model. However, if the sequence has low complexity, the logarithm model produces better results with any base layer rate.
4) The reasons behind 2) and 3) can be explained as follows. If the base layer rate is high or the sequence has low complexity, fewer bitplanes will exist in the enhancement layer. In this case, the linear relationship between the bitrate and percentage of nonzero-quantized coefficients is accurate as indicated by our results. This relationship is employed by the logarithm model, it is why it produces good results in this case. On the other hand, the square root model produces higher deviations in the least-significant bitplanes. And since we have a few total bitplanes, the impact of these high deviations will dominate, and the performance of the square root model will suffer.

   For high and medium complexity sequences and when the base layer rate is small, the converse of the above argument applies. That is, more bitplanes exist in the enhancement layer and the linear relationship used by the logarithm model becomes less accurate, reducing the accuracy of the logarithm model. At the same time, the effect of the deviations in the least-significant bitplanes of the square root model decreases as the number of bitplanes increases, making the square root model more accurate.
5) Regarding the range of applicability, the logarithm model always produces valid results at all possible bitrates. This is not the case for the square root model. As we explained in Section IV-A, the square root model greatly over estimates the quality for small quantization steps, approaching infinity for a quantization step of 1. Small quantization steps correspond to high bitrates. This over estimation of quality limits the applicability of the square root model to about 70% of the possible bitrates for low complexity sequences.
6) The time complexity of the square root model is smaller than the logarithm model. The logarithm model is slow because estimating the slope parameter of the linear relationship between the bitrate and percentage of nonzero-quantized coefficients is expensive. This is because the estimation

process requires bitplane truncations and computing the corresponding percentage of the truncated coefficients.

7) In general, the empirical and semi-analytic R-D models are more accurate than the analytic R-D models.
8) Te curve-fitting methods employed by the semi-analytic models are time-consuming and may occasionally result in inaccurate estimations for the model parameters.
9) Finally, the piecewise linear model is simple and fairly accurate, when compared to all other models.

B. *Recommendations for Choosing R-D models*

Our experimental results enable us to provide guidelines on selecting the most suitable R-D model for a target streaming system. We summarize our recommendations in the following.

- *Streaming stored video sequences and video-on-demand systems.* In these systems, the R-D model can be computed off-line and it is desired to be as accurate as possible. This is because a video sequence is usually streamed many times to clients with heterogeneous network connections. This implies that the cost of building an accurate R-D model will be amortized over many sessions. Therefore, we recommend using the piecewise linear model, where the model parameters will computed once and stored in a meta file with every video sequence.
- *Streaming in real-time and video conferencing.* Timing is critical in this case. In addition, real-time streaming usually happens for sports events, which have high complexity. Therefore, we recommend using the square root model because it has low time complexity and it is fairly accurate for medium and high complexity sequences. The square root model can further be accelerated by adopting a Laplacian, instead of Laplacian mixture, density function for modeling the DCT coefficients. Estimating the parameters of Laplacian density functions is straightforward and much simpler than the expectation-maximization method used with Laplacian mixture density functions. With Laplacian density, the square root R-D model is at least an order of magnitude faster than with Laplacian mixture density, while the accuracy of the resulting R-D model does not suffer much.
- *Broadcasting news and low-complexity video clips.* News clips usually have low complexity. We recommend using the logarithm model in this case because it more accurate than the square root model. The cost of constructing a piecewise linear model may not be justified in this case because news clips are usually broadcast a few times.
- *Streaming in distributed and peer-to-peer environments.* In these systems, there are potentially many servers. It would be very costly if we let every server independently construct the R-D model for the

same video sequence. Also, video sequences are streamed many times. Therefore, we recommend that only one server, e.g., the creator or the introducer of the sequence, builds a piecewise linear model for the sequence and attach it as a meta file with video data.

References

[1] H. Radha, M. Schaar, and Y. Chen, "The MPEG-4 fine-grained scalable video coding method for multimedia streaming over IP," *IEEE Transactions on Multimedia*, vol. 3, no. 1, pp. 53–68, March 2001.

[2] W. Li, "Overview of fine granularity scalability in MPEG-4 video standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 11, no. 3, pp. 301–317, March 2001.

[3] M. Dai and D. Loguinov, "Analysis of rate-distortion functions and congestion control in scalable Internet video streaming," in *Proc. of ACM International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV'03)*, Monterey CA, June 2003.

[4] M. Dai, D. Loguinov, and H. Radha, "Rate-distortion modeling of scalable video coders," in *Proc. of IEEE International Conference on Image Processing (ICIP'04)*, Singapore, October 2004.

[5] J. Sun, W. Gao, D. Zhao, and Q. Huang, "Statistical model, analysis and approximation of rate-distortion function in MPEG-4 FGS videos," in *Proc. of SPIE International Conference on Visual Communication and Image Processing (VCIP'05)*, Beijing, China, July 2005.

[6] M. Dai, D. Loguinov, and H. Radha, "Rate-distortion analysis and quality control in scalable Internet streaming," *IEEE Transactions on Multimedia*, 2007, to appear.

[7] H. Hang and J. Chen, "Source model for transform video coder and its application I: Fundamental theory," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 7, no. 2, pp. 287–298, April 1997.

[8] L. Lin and A. Ortega, "Bit-rate control using piecewise approximated rate-distortion characteristics," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 8, no. 4, pp. 446–459, August 1998.

[9] Z. He, Y. Kim, and S. Mitra, "Low-delay rate control for DCT video coding via ρ-domain source modeling," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 11, no. 8, pp. 928–940, August 2001.

[10] Z. He and S. Mitra, "A unified rate-distortion analysis framework for transform coding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 11, no. 12, pp. 1221–1236, December 2001.

[11] W. Ding and B. Liu, "Rate control of MPEG video coding and recording by rate-quantization modeling," *Transactions on Circuits and Systems for Video Technology*, vol. 6, no. 1, pp. 12–20, February 1996.

[12] T. Chiang and Y. Zhang, "A new rate control scheme using quadratic rate distortion model," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 7, no. 1, pp. 246–250, February 1997.

[13] X. Zhang, A. Vetro, Y. Shi, and H. Sun, "Constant quality constrained rate allocation for FGS-coded video," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 2, pp. 121–130, February 2003.

[14] Y. Wang, J. Ostermann, and Y. Zhang, *Video Processing and Communications*. Prentice Hall, 2002.

[15] ISO/IEC 14496-2, "Coding of audio-visual objects - part 2: Visual," June 2004.

[16] F. Wu, S. Li, and Y. Zhang, "A framework for efficient progressive fine granularity scalable video coding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 11, no. 3, pp. 332–344, March 2001.

[17] K. Matsuo, K. Takagi, A. Koike, and S. Matsumoto, “A bit-plane coding scheme of MPEG-4 FGS with high efficiency based on the distribution of significant coefficients,” in *IEEE Pacific Rim Conference on Multimedia (PCM’02)*, Hsin-Chu, Taiwan, December 2002.

[18] H.-Y. Chao, J.-S. Wang, J.-L. Lin, K.-C. Yang, C.-M. Wu, C.-M. Huang, and L.-D. Van, “High-performance low-complexity bit-plane coding scheme for MPEG-4 FGS,” in *IEEE International Conference on Multimedia and Expo (ICME 05’)*, Amsterdam, Netherlands, July 2005.

[19] G. Sullivan and T. Wiegand, “Rate-distortion optimization for video compression,” *IEEE Signal Processing Magazine*, vol. 15, no. 6, pp. 74–90, November 1998.

[20] W. Martinez and A. Martinez, *Computational statistics handbook with Matlab*, 1st ed. Upper Saddle River, NJ: Chapman and Hall, 2002.

[21] H. J. Park and T. W. Lee, “Modeling nonlinear dependencies in natural images using mixture of Laplacian distribution,” in *Advances in Neural Information Processing Systems (NIPS’04)*, Vancouver, Canada, December 2004.

[22] C. Hsu and M. Hefeeda, “Source models for fine-grained scalable video sequences,” Simon Fraser University, Tech. Rep., 2006.

[23] S. Mallet and F. Falzon, “Analysis of low bit rate image transform coding,” *IEEE Transactions on Signal Processing*, vol. 46, no. 4, pp. 1027–1042, April 1998.

[24] M. Dai, “Rate-distortion analysis and traffic modeling of scalable video coders,” Ph.D. dissertation, Department of Electrical Engineering, Texas A&M University, December 2004.

[25] Z. He and S. Mitra, “A linear source model and a unified rate control algorithm for DCT video coding,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 12, no. 11, pp. 970–982, November 2002.

[26] F. Muller, “Distribution shape of two-dimensional DCT coefficients of natural images,” *IEEE Electronics Letters*, vol. 29, no. 22, pp. 1935–1936, 1993.

[27] R. Joshi and T. Fischer, “Comparison of generalized Gaussian and Laplacian modeling in DCT image coding,” *IEEE Signal Processing Letters*, vol. 2, no. 5, pp. 81–82, 1995.

[28] K. Sharifi and A. Leon-Garcia, “Estimation of shape parameter for Generalized Gaussian distributions in subband decompositions of video,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 5, no. 1, pp. 52–56, February 1995.

[29] M. Varanasi and B. Aazhang, “Parametric generalized Gaussian density estimation,” *Journal of the Acoustical Society of America*, vol. 86, no. 4, pp. 1404–1415, 1989.

[30] C. Hsu and M. Hefeeda, “On the accuracy and complexity of rate-distortion models for fine-grained scalable video sequences,” Simon Fraser University, Tech. Rep. TR 2006-12, August 2006, available online at http://nsl.cs.surrey.sfu.ca/projects/fgs/.

[31] D. Adjeroh and M. Lee, “Scene-adaptive transform domain video partitioning,” *IEEE Transactions on Multimedia*, vol. 6, no. 1, pp. 58–69, February 2004.

[32] Web Page of Video Traces Research Group, Arizona State University, 2006, http://trace.eas.asu.edu/yuv.

[33] Web Page of Center for Image Processing Research, Rensselaer Polytechnic Institute, 2006, http://www.cipr.rpi.edu/resource.

[34] ISO/IEC 14496-5, “MPEG-4 Visual reference software,” February 2004.